RESEARCH MACHINES

COS 3.4
REFERENCE MANUAL

Contents

1	INTRODUCTION	1.1
	How to use this manual	1.1
	Underlining convention & the importance of RETURN	1.2
	A first look at the keyboard	1.2
	Entering hexadecimal numbers	1.2
2	MONITOR COMMAND LEVEL	2.1
	Disc support	2.2
	Cassette support	2.3
	Printer support	2.6
	Other monitor commands	2.8
3	FRONT PANEL	3.1
	Front panel display	3.1
	Commands for modifying memory	3.2
	Commands to modify registers	3.4
	Commands for accessing I/O ports	3.4
	Single step, break point, continue and jump	3.5
	Other front panel commands	3.6
	Entering the Front Panel from software	3.9
4	EMT TRAP CALLS	4.1
	EMTs returning control to COS command level	4.1
	Screen EMTs	4.2
	Keyboard EMTs	4.8
	General EMTs	4.11
	Printer support EMTs	4.15
	Disc EMTs	4.18
	Cassette EMTs	4.21
	I/O TRANSFER	4.24
	I/O channels	4.24
	I/O transfer vectors	4.25
	Non - I/O transfer vectors	4.25
5	PERIPHERAL INTERFACING	5.1
	User I/O port	5.1
	Keyboard	5.1
	The video display	5.2
	Cassette interface	5.4
6	MISCELLANEOUS FUNCTIONS	6.1
	Reset and initialisation	6.1
	Position independent code	6.1
	Interrupts	6.2
	Memory allocation and management	6.3
	Memory map	6.4

 Copyright October 1980

APPENDICES

A Quick Reference Guide A.1

B Changes from version 3.0 B.1

C Slow cassette support C.1

Index I.1

CHAPTER 1

INTRODUCTION

The RML Cassette Operating System (COS) consists of a resident 4K byte firmware monitor supplied in read-only memory. The monitor supports the RML video terminal, keyboard, cassette, disc and printer interfaces, implements a variety of 'front panel' functions, and supplies various useful subroutines. The most significant changes in the new release of COS 3.4 are an extensively modified screen handler, faster access to subroutines and an increased flexibility for tailoring these routines to fulfil alternative requirements.

HOW TO USE THIS MANUAL

If you plan to use the 380Z for programming in a High Level language only, you should read Chapter 2 for initialisation procedures. The later chapters are intended more for machine language programming support.

After power on or after a reset, COS will be in Monitor command level. In disc systems, at this stage, it is necessary to start the disc operating system and in cassette systems to load a program from cassette. It is also convenient to set up the appropriate handling routines for a printer interface. These functions, as well as commands setting cassette transfer rates and enabling a High Resolution Graphics board if installed as memory, are described in Chapter 2.

Typing CTRL F (holding down the CTRL key and F at the same time) displays the Front Panel on the screen. This shows the contents of the Z80 registers, the contents of memory addressed by them and the contents of a separate 40 byte block of memory. Whilst in Front Panel mode, register and memory contents can be modified at any time. The front panel is useful for debugging machine code programs, not only by displaying registers and memory, but also by the provision of a single step feature and the capability of inserting break points into programs. These features are described in Chapter 3.

Chapter 4 is a reference section for the large variety of useful routines available in COS. By a special mechanism, these are available to the user without having to know where they are located in memory in order to allow compatibility between future versions of COS. Additionally, if it is desired, these routines can be tailored to individual requirements by the use of transfer vectors.

Chapter 5 describes the working of the the screen, keyboard, user I/O port and cassette interface. Chapter 6 describes the use of COS in special cases, for example with interrupts and position independent code. It then provides a memory map of the 380Z, and instructs how best use of memory can be made.

For convenience, the major commands and routines available in COS are listed in Appendix A as well as the Quick Reference Guide, a copy which you can keep by the computer. For those who are already familiar with

 Copyright October 1980

previous versions of COS, details of the changes since COS 3.0 are summarised in Appendix B. Appendix C describes the slow cassette utility program. A listing of the COS monitor is also available on request, which can be used for aiding one's assembly language programming techniques.

UNDERLINING CONVENTION AND THE IMPORTANCE OF RETURN

Throughout this Manual, in the examples of the use of commands, the characters that you should type are underlined to differentiate them from the ones that the computer will print. For example, for

```
-L
Name>PROG
```

the computer prompts with '-', the user types 'L', after which the computer will follow with 'Name>'. The user then types 'PROG' followed by the RETURN key. It is important to emphasise here that single letter commands in COS, like L above, will function immediately and typing the RETURN key afterwards is unnecessary. However, any input of more than one number or character, like PROG above, must be terminated by RETURN.

A FIRST LOOK AT THE KEYBOARD

A number of versions of the 380Z keyboard have been produced. Most of these should include the important keys that will now be described. The keyboard is fairly similar to a typewriter one, but there are certain keys that have a special significance in COS commands:

RETURN — As described above, RETURN terminates an input of several characters, instructing the computer then either to execute the command or issue the next prompt.

CTRL — Holding down this key while typing one of the letter keys alters the letter key's action. Keys typed while holding down CTRL are referred to as the control keys and are prefixed with CTRL in the nomenclature used in this Manual. For example, CTRL F corresponds to holding down the CTRL and F keys, and is pronounced as 'control F'.

DELT — The DELT key, which stands for delete, rubs out the most recently typed character. Some versions of the 380Z keyboard however require that the SHIFT key be held down while typing DELT.

ENTERING HEXADECIMAL NUMBERS

A number of the Monitor and Front Panel commands require the user to enter a hexadecimal number. For example, the jump command (J) issues a prompt

```
-J>
```

and waits for an address to be entered, to which execution will be transferred. All such entries accept up to four hexadecimal digits (0-9, A-F) terminated by any non-hex character. If less than four digits are entered, leading zeroes are inserted on the left. Thus if 1A0 is entered as an address, 01A0 will be returned. The DELT key can be used allowing characters to be deleted, the most recently entered one first. If no characters are entered or all are deleted before the terminator, 0000 is returned. There are also commands which expect byte entries (two digits), in these cases, only the least significant (right hand) digits are used.

Although the entries can be terminated by any non-hex character, it is best to standardise on the RETURN key for conformity.

 Copyright October 1980

THIS PAGE INTENTIONALLY LEFT BLANK

 Copyright October 1980

CHAPTER 2

MONITOR COMMAND LEVEL

When power is first applied, the COS monitor prints its name and version number on the screen, followed by a right facing arrow:

```
COS 3.4/F
-
```

or something similar. The arrow is the Monitor prompt and indicates that the Monitor is expecting a command. The 'F' indicates that this is a Full disc version of COS, Mini disc versions will type 'M', Cassette versions 'C'. If there are other letters, these will represent revisions of COS. Whenever contacting Research Machines concerning COS, the whole version number displayed should always be quoted.

After switching on the 380Z or when commencing something new, it is necessary with cassettes to load a new program; how to do this is descibed under the heading 'Cassette support'. With discs, the user must load into memory the disc operating system; this is dealt with under 'Disc support'. As well as these, it may also be necessary to select a particular printer option, or enable a High Resolution Graphics board as memory. These operations are the type available in Monitor command level:

B	Bootstrap CP/M from disc and execute
X	Bootstrap CP/M from disc drive B
L	Load program from cassette
D	Dump memory to cassette
C	Continue program at restart address
J	Jump to address and execute program
O	Select printer option
M	Enable High Resolution Graphics board as memory
CTRL C	Return to Command Level
CTRL F	Enter Front Panel mode
CTRL Z	As CTRL F
CTRL T	Enter Typewriter mode

All COS commands, in both Monitor and Front Panel modes, may be in upper or lower case. File names entered in lower case are translated to upper case before being used.

CTRL F is the standard way to enter Front Panel Mode from both Monitor Command Level and from user programs. However, CTRL Z is also available for this from Monitor command level, in order to maintain compatibility with previous versions.

CTRL C is the prominent command for interrupting cassette procedures and returning to Command Level. However, this is not the case after CP/M has been read in, with either the B or X commands, when CTRL C, issued from

 Copyright October 1980

COS, will reenter CP/M by means of a 'reboot' operation. CTRL B is therefore made available when it is necessary to return to COS Command Level from the disc environment. This is done by first typing CTRL F from CP/M to enter Front Panel mode, then CTRL B to return to COS Command Level. This change in function of CTRL C is cancelled only by the Reset button.

CTRL T is the command which enables you to use the 380Z as a typewriter. After entering this command any character entered from the keyboard will be echoed onto the currently selected printer.

DISC SUPPORT

The following two commands will only work in disc versions of COS.

B Boot Disk Operating System

The B command starts the CP/M Disc Operating System from drive A. COS reads a sector from Drive 0, Track 0, Sector 1 into memory at 80 to FF hex, then jumps to it if there is no error. If the disc is off-line or an error occurs the message

Disc not ready

is issued and control returns to Monitor Command Level. If this occurs, check for an improperly inserted disc or, one without CP/M on, then retype B to repeat the bootstrap operation.

In order to return to CP/M from both Monitor command level and user programs, the standard command is CTRL C. As described above, CTRL B is available when it is necessary to return to COS command level in the disc environment.

X Boot off drive B

The X command causes COS to boot CP/M off drive B. This effectively swaps drives A and B and C and D as far as CP/M is concerned, so that all references to A: will access drive B, and vice versa. Some programs, however, contain their own disc handlers - examples are some versions of FORMAT and MOVCPM. Such programs revert to the normal state, so that for example references to A: access drive A. THIS IS ESPECIALLY HAZARDOUS WITH FORMAT.

In a disc system, B and X will normally be the only commands issued from Monitor Command Level. CP/M will automatically be cancelled after hitting the reset button, and either of the B or X commands will have to be reissued for any subsequent disc operation.

CASSETTE SUPPORT

L Load program from cassette

The L command is used for loading programs from cassette, the transfer rate initially being 1200 Baud or that selected (see below).

On typing L in response to the monitor prompt, COS requests the file name of the desired program or block of memory stored on cassette tape. For example, to load the program PROG, type:

```
-L
Name>PROG
```

The keys you must type are underlined - the rest are supplied by the computer.

A file name consists of from 1 to 6 alphanumeric characters (A-Z, 0-9). COS automatically adds trailing blanks (ASCII 20 hex) to make the name up to 6 characters. Terminate the file name with RETURN. File names entered in lower case are translated to upper case before being used.

After entering the desired file name, start the cassette recorder in PLAY. COS prints any file name it encounters, including the requested one and then starts loading it. Loading is indicated by blinking of the cursor and by a 4 digit hexadecimal number, which is the load address of the most recently read data block. Normally loading continues until an end record is read, when control is transferred to the address specified in the end record (the Monitor if address 0 is specified, else the program start address).

A greater than sign (>) is output after each record which is not loaded. These will be displayed during the passing of an unwanted file.

If an error is detected, COS prints

```
?Load err?
```

and waits for a response from the user. Two responses are possible:

1 Type CTRL C and start again.

2 The last address displayed (below the echoed file name) is the load address of the block in which the error was detected. Wind the tape back a short distance, then type L. Loading will be resumed and if the same (or an earlier) address is displayed, the tape has been wound back far enough. The block may now load correctly. If you cannot achieve a correct load after a few attempts, the tape may have become corrupted.

 Copyright October 1980

Loading can be interrupted at any time, and control returned to Monitor level, by typing any CTRL key. Note that keys such as RETURN and LINE FEED generate control codes and will abort a read operation.

A listing of all files on a cassette can be obtained by specifying a dummy or null file name in the L command.

D Dump memory to cassette

In order to save a program or block of memory, the D command is used. Like L, it will operate at 1200 Baud by default, and a 'Slow' speed of 300 Baud can also be selected in cassette COS with the O command.

On typing D, COS first requests a file name for the saved data. This should be entered as described for L. Then the limits of the area of memory to be saved are requested. The first and last addresses should then be entered in hexadecimal. The contents of locations between these limits inclusive, will be dumped. Finally, a start address is requested which will be the address to be transferred to after loading. Again the keys you type are underlined here, the rest are supplied by the computer:

```
-D
Name>PROG
First>100
Last>13A2
Start>100
```

Prior to terminating the start address with RETURN, start the recorder in RECORD mode and check that there is a cassette wound past the leader tape in the recorder. This is necessary as it is not possible to record on the non-brown leader tape. The dump is complete when COS prints an arrow.

COS will print

```
?Err?
```

if the first address is not lower than the last.

The D command can be interrupted as with L by any CTRL key, although there is a 5 second delay at the start and finish during which this will have no effect.

In the cassette environment, location zero contains a jump to COS command level. Thus address zero can be used as the start address of programs dumped on tape when they are not required to be self-starting. This should be used for example when loading data as opposed to programs.

O Set cassette transfer speed

All data transfers to and from cassette using the L and D commands will take place by default at 1200 Baud. However, a slower speed of 300 Baud to CUTS standards can be selected in cassette systems with the O option command. The 1200 Baud rate is refered to as 'Fast' and the 300 Baud rate as 'Slow'.

In order to select either speed, type the O command, followed by C:

```
-O
Cassette or lineprinter (C/L):C
Tape speed (FF/SS):
```

If typing O results in the message

```
Printer type (0-6):
```

you have a disc version of COS which no longer supports slow cassette and a separate utility program must be loaded first (see Appendix C).

In return to the prompt

```
Tape speed (FF/SS):
```

the desired tape speed should then selected by typing one of:

FF	Fast read and write
SS	Slow read and write
FS	Fast read, slow write
SF	Slow read, fast write

The rarely used FS and SF combinations are sometimes useful for tape conversion.

Software is normally supplied saved at 300 Baud for maximum safety. However, you should make fast copies immediately, locking away the 'slow' originals as masters. The format used for recording at 1200 Baud allows less scope for automatic error recovery than the 300 Baud one, and, in theory, read errors, due for example to uneven tape coating, are more likely to occur. However, given the use of reasonable quality tape, there should be very few problems with the faster format. As a comparison, Extended BASIC Version 5 loads in two minutes in 'fast', compared with about nine in 'slow'.

The new speed will remain in effect until either changed again or after a reset. The fast FF option will then come back into effect.

 Copyright October 1980

C Continue program at restart address

When a program is loaded from cassette tape, the program start address is saved at monitor location ADDR (see Monitor listing for the exact location of ADDR). Typing C in Monitor mode will cause a jump to the saved start address + 3.

By convention, RML system programs (e.g. BASIC) have a restart address 3 bytes above their initial start address. Thus typing C in these cases continues execution at the program restart address.

To allow use of the C command user assembly language programs should begin:

```
JP    START
JP    RESTART
```

if action should differ in the two cases, or with 3 NOPs if not.

If address zero is given as the start address when dumping on tape, and such a tape is subsequently loaded, the C command recognises this and jumps to location zero (instead of the jump to location 3).

In the disc environment, it is often not possible to restart programs. Therefore when CP/M is bootstrapped, the address used by the C command is reset to zero. A C command issued after this will cause CP/M to 'reboot' in the same way that CTRL C does.

PRINTER SUPPORT

COS provides the capability to redirect program output to a printer and to select a type of printer interface with the O command. Centronics and Anadex printers, which require a parallel interface via the User I/O Port, and printers requiring a serial interface (RS-232 or 20 mA) connected via the RML SIO-1, SIO-2, SIO-4, SIO-5 and SIO-6 serial interfaces are all supported.

Typing O in cassette versions of COS will result in the prompt

```
Cassette or lineprinter (C/L):
```

the reply of which should be 'L'. This will be followed in all versions by the prompt

```
Printer type (0-6):
```

to which the user should respond with one of the digits 0 to 6, chosen from the following list:

Code	Printer type
0	Console screen
1	SIO-1
2	SIO-2 or SIO-2B
3	Parallel (Centronics or Anadex DP8000 and DP9501)
4	SIO-4 (Qume Sprint 5)
5	SIO-5
6	SIO-6

Users owning a Research Machines Microline 80, may have any interface, either parallel or serial. If it uses a serial interface, a Baud rate of 2400 should be selected (see below).

If in doubt as to which interface code to select, check the board inside the computer which connects directly to the socket on the back panel. If it is connected to the board by a flat 20 way cable, it is a parallel interface, type 3. If the board is a small one, about three inches by two, it is either an SIO-2 or 2B, and therefore type 2. If it is a standard sized card (about 10"x5") with about a third of the board completely blank, it is a type 1 interface. Otherwise it is type 4.

When appropriate, a Baud code will also be required by the prompt:

```
Baud code (0-6):
```

The code should be selected from:

Code	Baud rate
0	110
1	300
2	600
3	1200
4	2400
5	4800
6	9600

The printer will probably print garbage if the wrong Baud code is selected.

The default for this command is type 0, the screen. All output will revert to this if the reset button is pressed after a printer selection.

Printer type 1 is used with SIO-1 and SIO-1B serial interfaces, which are identical from a software viewpoint. The Baud rate for this type of interface is selected by means of switches on the interface card itself, so no Baud code is required.

Printer type 2 is used with SIO-2 and SIO-2B serial interfaces. These require a Baud code, but, because these devices will not operate beyond 2400 Baud, the maximum Baud code allowable is code 4.

Printer type 3 is used for printers with a parallel interface, such as most Centronics and Anadex models. Being parallel, Baud rate has no

meaning, and printer type 3 therefore requires no Baud code. The printer will be placed on-line automatically at the time it is selected. If the message

Attend to printer!

appears, the printer failed to respond correctly to the 380Z. Usual causes are:

Printer not switched on.
Printer plugged into the wrong socket on the 380Z.
Wrong type of connecting cable used.
Printer out of paper.
Printer not selected by select/on line switch on printer.

Printer type 4 is used with an SIO-4 serial interface. This type of interface will handle any Baud rate up to 9600, so allowable Baud codes range from 0 to 6.

Printer types 5 and 6 are additional serial interfaces identical to the SIO-4. The only difference between these types is the I/O ports at which they are located. These are:

	FDS	MDS	Cassette
Suffix	F	M	C
Type 4	E8	C8	C8
Type 5	C8	E8	E8
Type 6	48	48	48

In all cases the system will appear to 'hang-up' if the printer goes off-line during a printing run. Either correct the problem or type CTRL C to abort. If the error was correctable, for example, reaching the end of the ribbon, none of the printer output will be lost during this period, and can be continued as if nothing had happened.

OTHER MONITOR COMMANDS

J Jump to address and execute program

Programs, especially ones that are not self-starting, can be executed with the J command. On typing J, COS requests an address (in hexadecimal):

-J>100

Type an address, followed by RETURN, to jump to the address and begin execution.

M Enable High Resolution Graphics board as memory

The M command enables the High Resolution Graphics board as memory and restarts COS. This command is more or less equivalent to pressing the reset button, in that it rewrites all the low and high memory scratchpad normally written on reset. This command should be issued only in a 32K system, without any add-on RAM.

 Copyright October 1980

CHAPTER 3

FRONT PANEL

Typing CTRL F from Command Level and most programs, displays the 'Front Panel' on the screen. This shows the contents of the Z80 registers, the contents of memory addressed by them and the contents of a separate 32 byte block of memory. Whilst in Front Panel mode, register and memory contents can be modified at any time. Additionally, programs can be debugged with the aid of commands that start, single-step through, and insert breaks into programs. If only slight changes are being made in the Front Panel, such as setting a printer option, the program can be reentered with no loss of context.

The commands available are:

RETURN	Increment memory pointer
-	Decrement memory pointer
LINE FEED	Advance memory pointer by 8
/	Move memory pointer back by 8
CTRL L	Advance memory pointer by 32
CTRL O	Move memory pointer back by 32
M	Set memory pointer
I	Set memory pointer from memory contents (absolute)
R	Set memory pointer from memory contents (relative)
U	Set memory pointer from PC contents
.	Increment register pointer
X	Switch register display to alternate set
V	Update PC from memory pointer
G	Get first occurrence of specified pattern of bytes
N	Find next occurrence of pattern
H	Hexadecimal calculator
P	Fill and test memory between limits
S	Shift memory content
O	Set printer option
J	Jump to address and start execution
K	Continue program execution from PC
Z	Single step
<	Increment I/O port pointer
>	Decrement I/O port pointer
,	Write byte to I/O port
CTRL B	Return to monitor command level
CTRL C	Go to either command level or to CP/M if booted

FRONT PANEL DISPLAY

Front Panel mode can be entered from monitor mode by typing CTRL F in response to the COS monitor's arrow, or under software control. In this

mode, the screen scroller acts only on the bottom four lines of the screen, while the upper twenty lines are used to display register and memory contents. Whilst in front panel mode, COS prompts with an exclamation mark (!).

The upper part of the Front Panel display shows the current contents of the Z80 registers and of memory locations addressed by them. The display relating to each register occupies a single row; registers are identified by their standard Zilog abbreviations, namely, from the top of the screen downwards by PC, SP, IY, IX, HL, DE, BC, AF, referring to the program counter, the two index registers, the three 16-bit registers, the accumulator and the flag register respectively. When the alternate set of registers is displayed, their names are tagged with an apostrophe, that is, HL', DE', BC', AF'. Down the left hand edge of the register area, an arrow indicates the register or register pair currently selected for modification, initially the program counter.

Immediately to the right or the register name, its content is displayed as a four digit hexadecimal number, and to the right again are shown 8 bytes (2 digit hex numbers) showing the content of the memory region addressed by them.

The byte actually addressed by the register or register pair is in the column with the upwards pointing arrow at the foot. To the left, in a given row, are the contents of addresses lower (numerically less) than the current register content, and to the right are the contents of higher locations. Thus, considering the program counter, four bytes following it are displayed. In the case of the accumulator, such a display would be meaningless; instead the content of the flag register F is displayed using the letters S, Z, H, V, N, and C to indicate the presence of any of these flags. Thus the appearance of Z means that the zero flag is set.

The lower part of the Front Panel display is entirely separate from the register display and shows the contents of an area of memory approximately centred on an address which can readily be changed, and which is indicated by two horizontal arrows. For each location both the address and content are shown.

The row between the register and memory displays shows the contents of eight I/O ports centred on the port address given at the left (initially port 0).

COMMANDS FOR MODIFYING MEMORY

Commands affecting the memory display in the lower half of the Front Panel are used both to examine memory content and to change it. When in Front Panel mode, COS checks each entry, first to determine if any hexadecimal digits have been entered and second to see what action the terminating non-hex character implies. Terminating characters which affect the memory display and content of memory do so with reference to the arrowed location in the second column of the memory display. This is referred to as the memory pointer location. If a hexadecimal byte precedes the terminating character it always replaces the current contents of the pointer location.

Certain terminating characters cause the memory pointer to be updated. The RETURN key increments the pointer to the next higher address; minus (-) decrements it. LINE FEED adds eight to the memory pointer while slash (/) subtracts eight from it. The memory pointer can be increased by 32 with CTRL L or decreased by 32 with CTRL O, thereby moving the whole memory block being displayed. Terminate a hexadecimal number by space if you do not wish to alter the memory pointer but do wish to update memory. If no hexadecimal number precedes the terminator the memory pointer is updated without affecting the content of memory.

The memory pointer can also be set using several other commands.

M Set memory pointer

It is possible to set the memory pointer to any desired location by means of the M command. When M is typed COS prompts for a four digit hex address which becomes the address of the memory pointer. Thus to set the memory pointer to 4400, type

!M>4400

followed by RETURN.

I Set memory pointer from memory contents (absolute)

The I command sets the memory pointer to the 16-bit address at the current location of the memory pointer. To use the I command, use the pointer commands described above to set the pointer to the lower byte of the address, then type I and that address will become the memory pointer address.

Thus, assuming that you are looking at the machine code for CALL 2200, which would appear in the memory display pointed to by the pointer as CD 00 22, type RETURN to advance the pointer to point to 00 (the lower byte of the 16-bit address), and then type I. The memory pointer will now point to address 2200.

Note that the I command expects the address low byte first in exactly the same way as the Z80 does.

R Set memory pointer from memory contents (relative)

In a similar manner to I, the memory pointer can be updated to a relative address with the R command; in this case it should be set to the byte containing the relative distance to the target address (for example, the second byte of a relative jump instruction) and then R should be pressed.

This works in the same way as the Z80 in that the command calculates the new address as address of relative instruction + displacement + 2.

 Copyright October 1980

U Set memory pointer from PC contents

The memory pointer can additionally be set from the contents of the program counter PC as displayed at the top left hand corner of the front panel display. This is useful during program debugging to display op-codes to be executed in the memory block.

COMMANDS TO MODIFY REGISTERS

The contents of the register or register pair selected by the arrow at the left hand edge of the register area can be modified by entering a hexadecimal number terminated by a period or full stop (.). Note that in this case, in distinction to modification of memory content, a 16-bit value (4 hexadecimal digits) is expected. Thus if register BC contains 0012 and you wish to change the contents of register B from 00 to 34, 3412 must be entered. After changing the contents of a register, that register remains selected by the register pointer; however if period alone is typed without a preceding hex number, the register pointer steps on to the next register. Thus any register can rapidly be selected.

X Switch register display to alternate set

When front panel mode is entered the register set displayed is that which was in use at the moment of entry. The alternate set of registers (HL′, DE′, BC′, AF′) can be displayed instead at any time by typing X or W. This switches all operations on the registers to the alternate set and that set remains in use until X or W is typed again. If Front Panel mode has been entered by a software call, a break point for example, and the program is later allowed to continue, the set of registers returned to the program will be that currently displayed. It is necessary to keep track of which set is in use at any time but this is quite easily done as, when the alternate set is being displayed, their names are tagged with an apostrophe.

W, which also functions as X does, is provided in order to remain compatible with previous versions of COS which used W. X was later introduced for convenience.

V Set PC from memory pointer

The contents of the displayed program counter can be directly updated with the memory pointer address using the V command.

COMMANDS FOR ACCESSING I/O PORTS

The row between the register and memory displays shows the content of eight I/O ports centred on the port address given at the left. The latter may be altered, as for the registers, by entering the desired port number, terminated by a full stop, when the register selection pointer is pointing to the I/O row.

Alternatively, the characters < or > may be typed, incrementing or decrementing the I/O port pointer respectively (imagine that they move the row of port contents to the left or right).

To write a value to the current port (i.e the port whose number appears immediately to the right of the I/O legend and whose content is indicated by an upwards facing arrow) type a hex value, terminated by a comma. If no value precedes the comma, it is ignored.

SINGLE STEP, BREAK POINT, CONTINUE AND JUMP

Z Single step

Whilst in Front Panel mode it is possible to execute a program a single instruction at a time. The desired starting address is first entered into the program counter as described above, then the Z key is pressed once for each instruction. After the execution of an instruction the register and memory display is updated to reflect any changes that have occurred.

Note that some of the more complex Z80 instructions, such as LDIR, require multiple 'single steps' for completion. Thus if an LDIR instruction is set up to do eight transfers, eight depressions of the Z key will be necessary. This is because of the way the Z80 executes such instructions. A further difficulty is that it is not really possible to single step through routines which make use of external timing signals.

The single step takes place as a result of an NMI occurring during the target instruction but immediately prior to the instruction the contents of locations 0066 to 0068 hex are saved and restored immediately afterwards. For this reason it is not possible to single step correctly through these locations.

Break points

A convenient solution to these difficulties is to make use of a 'break point'. The first byte of an instruction following the time critical section of code is replaced by the break code (FF hex) by using the modify memory commands. The program is then allowed to execute in the normal way using K (see below). When the break point code is encountered, 'BREAK' is displayed and front panel mode is entered with the machine state that was present just prior to the execution of the break instruction. The program counter and memory pointer will both contain the address of the break instruction. That address can now be modified (using the modify memory commands) to its original contents and the program executed in single step mode. The register and memory contents can of course be altered before doing so.

K Continue program execution from PC

If it is desired to continue the program at normal execution speed, perhaps having first installed further break points in the program, this can be done with the K command which restores all registers and continues execution at the address currently in the program counter. The contents of the program counter can be modified before typing K to continue at a different address or alternatively, the J command can be invoked.

J Jump to address and execute program

As mentioned above, the J command can be utilised for starting programs without initially setting the program counter to the start address. On typing J, COS requests an address (in hexadecimal):

```
!J>100
```

Type an address, followed by RETURN, to jump to the address and begin execution. Entering the address in the J command can be cancelled by typing CTRL F.

OTHER FRONT PANEL COMMANDS

G Get first occurrence of specified pattern of bytes
N Get next occurrence of pattern

Two commands are available in Front Panel mode to enable a search to be made of memory for a specified pattern of bytes. The G command allows the entry of a series of bytes and then searches memory, starting at the byte following the current memory pointer location for the occurrence of that pattern. The search stops when the pattern is found and the memory display is updated, the memory pointer location containing the first byte of the sought pattern. The search can be continued to look for further occurrences of the pattern by typing N. Searching continues from the byte following the current memory pointer address and stops with the pointer at the first byte of the next occurrence of the pattern. You can start the search at any desired location by setting the memory pointer to that location before typing N.

After typing G, the pattern is entered a byte (two hex digits) at a time, terminated by RETURN. A null entry (RETURN only) terminates the pattern and initiates the first search. For example to find the pattern CD0022 hex, type:

```
!G
>CD
>0
>22
>
!
```

The pattern can be any reasonable length from one byte upwards. It is stored in the area of random access memory at the top of memory dedicated to the Monitor between the Monitor transfer vectors and the Monitor stack. The search function is usually used with strings of up to three of four bytes. For example it is valuable for relocating calls from one address to another. If N is typed several times, or if the entered pattern is not present in memory, the search will terminate with the memory pointer at the address where the pattern is stored. This is at label PATN (see the Monitor listing for the exact address of PATN) and the termination of a search at this address is a convenient indication that no further occurrences of the pattern are present. The stored pattern will be cancelled if the hexadecimal calculator routine H (see below) is used before an N command.

H Hexadecimal calculator

Typing H enters the hexadecimal calculator routine. COS prompts for two hexadecimal numbers (each terminated by RETURN) and displays their sum and difference. This is useful for example for working out relative addresses. For example to find the sum and difference of 4A and 20 hex, type:

```
!H
>4A
>20
006A 002A
!
```

P Fill and test memory

Typing P invokes a function which allows the user to fill a section of memory with a selected byte. COS prompts for the first and last addresses to be filled, and then for the byte with which to fill them. All locations between the specified limits, inclusive, will be filled with this byte. Thus to clear the whole of user memory in a 16K system, enter 100 and 3BFF as the first and last addresses, and 0 as the byte to fill them with:

```
!P
First>100
Last>3BFF
With>0
!
```

 Copyright October 1980

It is possible, although not very useful, to fill a single location by specifying the same location for the first and last address, but COS prints

?Err?

if the last address is lower (numerically less) than the first.

While memory is being filled, COS reads it back to make sure that the new contents have been correctly stored. This provides a simple check of memory integrity. If a location does not read back correctly,

?Mem err?

is displayed and the Front Panel display is updated such that the memory pointer indicates the location at which the error has been detected.

One can use this command as a quick test to find out how much memory is available in a system by deliberately specifying an excessively high address. ?Mem err? will be displayed and the last available memory location will be the one just before the memory pointer. Note that errors will be produced by non-existent memory, which reads as FF, by video display memory, where reading to is normally disabled, and by the COS ROM between E000 and EFFF, being read only.

S Shift memory content

In order to shift memory, type S, after which COS will prompt for the first and last addresses of the block of memory which is to be moved, and then for the first address of the area to which the block is to be moved to. Enter all three addresses as 16-bit values (four hex digits). It is possible to move a block of any size; COS types

?Err?

if the address of the first location of the block to be moved is higher than the last. A block of memory can be moved in either direction, that is to a higher or lower address than it is presently situated at and the new area may overlap the old. COS automatically uses the appropriate algorithm to ensure that the block is transferred correctly.

O Select printer option

The O command for selecting printer options and also cassette speeds in cassette versions of COS is available in Front Panel mode as well as Monitor command level. Its operation is identical to that described in command level.

CTRL B Return to Monitor command level
CTRL C Go to either command level or CP/M if booted

In the cassette environment, both CTRL B and CTRL C will enter monitor command level. This will be necessary for example to load or dump a program on cassette. However, in the disc environment, CTRL C will enter CP/M command mode, and CTRL B will have to be used for going to Monitor command level.

ENTERING THE FRONT PANEL FROM SOFTWARE

User programs may enter the Front Panel in three ways:

1 Following the execution of FF hex, via the break point.

2 In keyboard EMTs, especially EMT KBDTF, after typing CTRL F.

3 Via a defined address which can be 'CALLed'. This entry address is currently at E013 hex but may change from time to time.

In cases 2 and 3, as long as the program counter PC is not changed, the K command will return to the instruction following the entering instruction without any loss of context.

 Copyright October 1980

THIS PAGE INTENTIONALLY LEFT BLANK

CHAPTER 4

EMT TRAP CALLS

Almost all accesses to the subroutines provided by COS are made via the EMT mechanism. This is a two byte call by number, that is the second byte gives the code number for a particular routine. On initialisation, COS places a jump instruction at address 30 hex. Whenever a program executes a RST 30H instruction, the EMT, COS looks at the following byte and uses this to decide which routine within the Monitor to execute. This process emulates in software the trap mechanism available on some computers, hence EMulator Trap.

The enormous advantages gained by using the EMT are:

1 Programs written for one version of COS will still run with other versions, since the code for each routine will be unchanged. If a simple call instead was made, the location of the subroutine could not be guaranteed in subsequent versions, thereby requiring significant changes to all machine code programs.

2 The external definition of an EMT will not normally be changed between versions of COS, so that if there is an alteration in the hardware of the system, the EMT internally can cater for the new requirements. Programs using EMTs will not have to be changed the slightest.

For these reasons, users are discouraged from making direct calls to COS routines. However, being an emulated instruction, it takes a relatively long time to execute (about 65us in versions 3.3 and later). This delay can be of importance when writing time-critical programs.

The trap calls are numbered from -1 upwards. RML may add new calls in future releases of COS, and these will be numbered upwards from the current limit.

Therefore, the EMT OUTC with code 1 would have an op-code of F701 hex.

EMTS RETURNING CONTROL TO COS COMMAND LEVEL

The following two EMTs are not really calls as such since they do not return to the calling program.

ERROR Print ?Err?, return to command level

This trap call types out the message ?Err? on the console and returns control to the main COS command loop.

Code: -1

 Copyright October 1980

CONTC Return to command level

Control is transferred to the main COS command loop.

Code: 0

SCREEN EMTS

The EMTs used for accessing the screen should be considered as divided into three types, those that send characters to the scroller, those that send characters anywhere on the screen, and those that provide special screen functions.

Mnemonic	Code	Hex	Function
OUTC	1	1	Output byte in reg A to screen
OPNWT	11	B	Open screen in frame blanking
CLOSE	12	C	Close screen
GRAFIX	13	D	Clear screen, 4 line scroller
SCROLL	14	E	Full screen scroller
CLEAR	15	F	Clear selected screen area
OUTC	22	16	As EMT 1 (not via transfer vector)
MSG	23	17	Output message at (HL) to screen
OUTNC	42	2A	Scroller output without clearing page key
OUTF	43	2B	Output to screen at (HL) from reg A
INF	44	2C	Input from screen at (HL) to reg A

In future versions of COS, the EMTs GRAFIX, SCROLL, CLEAR, OUTF and INF may be superceded, and therefore may not be fully compatible.

OUTC Send character to the screen at the current cursor position

This routine outputs the character given in the A register to the screen at the current cursor position. The cursor is incremented; if this takes the cursor off the right hand end of the screen, the cursor moves to the next line. If this takes the cursor beyond the bottom line, the screen contents are scrolled upwards one line and the cursor re-appears at the left hand end of the now empty bottom line. All characters with ASCII code greater than or equal to 20 hex, apart from DELT (ASCII 7F hex), are displayed; this includes digits 0 - 9, punctuation, the upper and lower case alphabets, and the special 380Z graphics set (codes 80 to FF hex). Most of the control codes (ASCII code less than 20 hex) are interpreted as cursor controls. Unless otherwise specified, control characters are non-destructive - that is, the character at the new cursor position will reappear if the cursor is moved again with another cursor control character.

The control codes are summarised here:

 Copyright October 1980

Character	Code	Hex	Description
CTRL D	4	4	Resume output (after CTRL O)
CTRL H (BACKSPACE)	8	8	Cursor left (non-destructive)
CTRL I (TAB)	9	9	Horizontal tab (non-destructive)
CTRL J (LINE FEED)	10	A	Cursor down
CTRL K	11	B	Cursor up
CTRL L (FORM FEED)	12	C	Clear screen, cursor to bottom left
CTRL M (RETURN)	13	D	Carriage return/line feed
CTRL N	14	E	Carriage return
CTRL O	15	F	Suppress output
CTRL Q	17	11	Clear paging counter
CTRL S	19	13	Set paging counter
CTRL U	21	15	Blink on
CTRL V	22	16	Initiate XY addressing
CTRL W	23	17	Blink off
CTRL X	24	18	Cursor right (non-destructive)
CTRL Y	25	19	Clear to end of line
CTRL]	29	1D	Cursor home (top left)
CTRL ¢	30	1E	Clear to end of line
CTRL _	31	1F	Cursor home and clear screen
DELT (RUBOUT)	127	7F	Destructive backspace

CTRL D — Resume output

This character causes output to resume. This overrides the effect of a previous CTRL O (see below).

CTRL H (BACKSPACE) — Cursor left

The cursor moves left one position. If the cursor was at the beginning of a line, it moves to the end of the previous line, unless the cursor was already on the top line, in which case it will move to the end of the top line.

CTRL I (TAB) — Horizontal tab

The cursor moves to the right to the next tab position. These are evenly spaced across the screen at intervals held in the location TABSTP in the monitor scratchpad. The exact address of TABSTP is given in the Monitor listing. Normally, tab stops will be at every eighth character position, starting at position 0.

CTRL J (LINE FEED) — Cursor down

The cursor moves down one line. If this moves the cursor off the bottom of the screen, the screen will be scrolled up and the cursor left positioned at the bottom left of the screen.

 Copyright October 1980

CTRL K — Cursor up

The cursor is moved up one line, unless it is already on the top line, in which case nothing happens.

CTRL L (FORM FEED) — Clear screen, cursor to bottom left

The screen is cleared and the cursor positioned at the bottom left of the screen.

CTRL M (RETURN) — Carriage return/linefeed

The cursor moves to the start of the next line; if the cursor was on the bottom line the screen will scroll upwards.

CTRL N — Carriage return

The cursor moves to the start of the current line.

CTRL O — Suppress output

After a CTRL O is sent to the screen, no further characters sent to the screen via EMT OUTC will appear until output is resumed with CTRL D.

CTRL Q — Clear paging counter

CTRL Q clears the paging counter, disabling the normal auto-paging feature. Auto-paging is described later.

CTRL S — Set paging counter

This character sets the paging counter, enabling auto-paging. Autopaging is described later.

CTRL U — Blink on

This character starts the cursor blinking. Whenever COS is waiting in the keyboard EMTs KBDW and KBDWF, the cursor position is blinked between the normal solid white cursor and the character underneath it. At all other times, there is no cursor visible. Note that most RML programs and virtually all CP/M programs take input via KBDWF, and a blinking cursor is thus feasible. However, TXED is almost impossible to use without the cursor steady.

CTRL V — Initiate XY addressing

CTRL V initiates X-Y cursor addressing. The following two characters define the vertical and horizontal screen position to which to move the cursor. Each has 32 decimal added to it, so that (32,32) is the top left of the screen and (55,71) is the bottom right. Once a CTRL V has been received, the cursor addressing

sequence must be completed.

CTRL W — Blink off

This character disables the cursor blinking enabled with CTRL U.

CTRL X — Cursor right

This moves the cursor right one space. If this moves the cursor beyond the right of the screen, the cursor will move to the start of the next line, and the screen will be scrolled if necessary.

CTRL Y — Clear to end of line

The screen is cleared to the end of the current line and the cursor position is left unchanged.

CTRL] — Cursor home

The cursor is moved to the top left hand corner of the screen (the home position).

CTRL ↑ — Clear to end of screen

This character clears the screen below and to the right of the cursor. The cursor position is left unchanged.

CTRL _ — Cursor home and clear screen

The screen is cleared and the cursor left positioned at the top left corner of the screen.

DELT (RUBOUT) — Destructive backspace

The cursor moves to the left one position and the character at that position disappears. If the cursor is already at the left hand edge of the screen, the character at the cursor position is cleared and the cursor does not move.

All other control codes are ignored.

The LINE FEED character is ignored if it immediately follows a RETURN. Thus, either RETURN or RETURN/LINE FEED will act as a newline code. Special note is taken when the 40th character is put on a line. The cursor moves to the next line, and the following character is ignored if it is a RETURN. This means that lines 40 characters long will appear on successive screen lines.

After a page-full of information has been sent to the screen, the cursor will start blinking between the normal solid white cursor and a hollow cursor, and output will stop. Output will

resume when any key is typed on the keyboard, for one page. CTRL A typed on the keyboard, at any time when a program expects input, will toggle the paging counter, enabling or disabling auto-paging.

This EMT is called via a transfer vector - see the monitor listing for the exact address. This means that alternative handlers can be installed, simply by adjusting the transfer to point to the new routine.

Registers affected: None

OPNWT Open screen in frame blanking

OPNWT is used to open the screen for access by programs. It waits for the beginning of a frame blanking period, and then opens the screen. Note that the screen will go blank when the screen is open, so it should only be opened during blanking. See the description of the screen in Chapter 5 for details of frame blanking.

Registers affected: None

CLOSE Close screen

The screen memory is closed for access by programs, allowing the screen to display the memory. See the screen description for details.

Registers affected: None

GRAFIX Clear screen, 4 line scroller

EMT GRAFIX clears the top 20 lines of the screen and confines scrolling to the bottom four lines. If the cursor was somewhere in the top part of the screen, it is moved vertically downwards until it is in the bottom part. Auto-paging still assumes that the screen is 24 lines long, so it should probably be disabled. However, this must be done separately.

Registers affected: None

SCROLL Full screen scroller

EMT SCROLL restores the full screen scroller after an EMT GRAFIX, allowing a full 24 lines of text. The cursor is unaffected.

Registers affected: None

 Copyright October 1980

CLEAR Clear selected screen area

A call to CLEAR allows any area of the VT screen to be cleared. Because the routine makes no checks on the validity of its arguments, it is necessary to specify the area to be cleared with great care, otherwise the results will be unpredictable.

On entry, register A contains the number of lines to clear. This should lie between the limits 1 and 24 (decimal) inclusive.

Register HL contains the address of the beginning of the top line of the area to clear. HL should contain a valid screen address and should point to the first character position of a row, that is, the six least significant bits of L must all be 0.

Typical parameters used with the CLEAR routine are:

A decimal	HL hex	Result with EMT CLEAR
24	F000	Clear the whole screen
4	F500	Clear the bottom 4 lines

On return to the calling program, register A contains 0 and register HL points to the address following the last character position cleared. This will be in a part of the screen memory which is not mapped directly to the screen. The character positions that have been cleared have had a graphics blank character (ASCII 80 hex) written into them. Registers B and DE are destroyed.

Registers affected: A,B,DE,HL and the flag register.

OUTC As EMT 1 (not via transfer vector)

EMT 22 outputs characters to the display scroller, exactly as for EMT 1. However, this trap call goes directly to the routine in the monitor and not via the VTV transfer vector. This EMT will be withdrawn.

Registers affected: None

MSG Output message at (HL) to screen

The MSG call is used to output a string of characters to the display scroller at the current cursor position. Register HL should point to the start of the character string. The end of string consists of a negative byte, with the most significant bit set, and this byte is not printed.

Registers affected: None

 Copyright October 1980

OUTNC Scroller output without clearing page key

OUTNC is the same as OUTC (code 1), except that any character apart from CTRL A typed to obtain a new page of text when auto-paging is active is not cleared. This means that it is available to any user program. A side effect is that if the program producing the output does not test the keyboard and clear the latch if necessary, OUTNC will not halt at the end of each page, because of the character held in the latch.

Registers affected: None

OUTF Output to screen at (HL) from register A

The character in register A is written to the memory address contained in register pair HL during the line blanking period with the screen opened for access by the processor. It would normally be used to write to the screen in such a manner that no flicker or 'snow' appears on the screen. See the description of the screen for details of screen timing and addresses.

Registers affected: None

INF Input from screen at (HL) to register A

This EMT reads the memory location whose address is in register pair HL and places the result in register A. The memory read is performed during line blanking with the screen opened for access by the processor, so that no flicker or 'snow' appears on the screen. See the description of the screen for details of screen timing and addressing.

Registers affected: A and the flags

KEYBOARD EMTS

All keyboard EMTs return information in the A register except KBDTF which leaves it unchanged. The KBDT group do not clear the keyboard, merely return status information. KBDW and KBDWF do not return until a character has been typed, whereas KBDIN returns whether one has been typed or not, and all the latter group clear the keyboard before returning. KBDW, KBDWF and KBDTL process CTRL A and are meant for user programs. KBDIN is primarily for internal use by COS.

The following table summarises the keyboard related EMTs:

Mnemonic	Code	Hex	Function
KBDC	2	2	Read keyboard, trap CTRL C
KBDIN	29	1D	Read and clear keyboard
KBDTC	30	1E	Test keyboard, return character
KBDTL	31	1F	Test keyboard, return -1 (true)
KBDTF	32	20	Test keyboard for CTRL F
KBDW	33	21	Wait for character from keyboard
KBDWF	34	22	Wait for key struck, check for CTRL F

KBDC Read character, trap CTRL C

This 'keyboard in' routine reads the keyboard interface. A character, if available, is returned in register A and the Z flag is cleared. If no character is available, zero is returned in register A and the Z flag is set. Note that KBDC returns immediately in either case. KBDC traps CTRL C typed at the keyboard, sending control to the COS command level in a cassette environment, or rebooting in a disc environment.

This EMT is called via a transfer vector - see the monitor listing for the exact address. This means that alternative handlers can be installed, simply by adjusting the transfer to point to the new routine.

Registers affected: A and the flags

KBDIN Read and clear keyboard

KBDIN reads the keyboard, and clears the keyboard latch if a character was typed. This routine will return the Z flag set and zero in register A if no character is available, else the Z flag reset and the character in A. KBDIN does not trap any character, so all are available to the user program.

Registers affected: A and the flags

KBDTC Test keyboard, return character

KBDTC reads the keyboard without clearing it. If a character is available, the Z flag is reset and the character returned in register A. Otherwise, zero is returned in A and the Z flag is set. CTRL A is processed, toggling the auto-paging flag.

Registers affected: A and the flags

KBDTL Test keyboard, return -1 (true)

KBDTL is the same as KBDTC, except that if any key apart from CTRL A is typed, KBDTL returns -1 (FF hex) in register A with the zero flag reset, else 0 in A and the zero flag set. CTRL A toggles the auto-paging flag and returns zero.

 Copyright October 1980

This EMT is called via a transfer vector - see the monitor listing for the exact address. This means that alternative handlers can be installed, simply by adjusting the transfer to point to the new routine.

Registers affected: A and the flags

KBDTF Test keyboard for CTRL F

KBDTF is a keyboard test for CTRL F. If this character is typed, COS enters the Front Panel (q.v.) Otherwise, it returns without clearing the keyboard latch and without affecting any registers. The only exception is that CTRL A toggles the auto-paging flag and causes the keyboard latch to be cleared.

If CTRL F is typed, the keyboard latch is cleared. If Front Panel mode is terminated with K, execution of the program which called KBDTF will be resumed with no loss of context.

Registers affected: None

KBDW Wait for character from keyboard

KBDW waits for a character to be typed on the keyboard. CTRL A is processed, toggling the auto-paging flag if typed. If the VT scroller has been set up with a blinking cursor (see EMT OUTC, code 1), KBDW will flash the cursor while waiting for a key to be typed.

KBDW returns the typed key in A with the Z flag reset. If CTRL A is typed, zero is returned in A with the zero flag set.

Registers affected: A and the flags

KBDWF As KBDW, trap CTRL F

KBDWF is the same as KBDW, except that it interprets CTRL F, entering the Front Panel when this character is typed. Program context is not lost, so that leaving the Front Panel with the K command reenters the calling program. Like KBDW, CTRL A is interpreted, toggling the auto-paging flag, and the cursor will blink during a call to KBDWF if the screen has been set up to use a blinking cursor - see EMT OUTC.

This EMT is called via a transfer vector - see the monitor listing for the exact address. This means that alternative handlers can be installed, simply by adjusting the transfer to point to the new routine.

Registers affected: A and the flags

GENERAL EMTS

Mnemonic	Code	Hex	Function
WAIT	16	10	Wait B 833 usec intervals
UPDATE	18	12	Copy MASK to PORT0
GETHEX	19	13	Get hex number from kbd in reg HL
DEOUT	20	14	Put reg DE to (HL) in ASCII
BYTEO	21	15	Put reg A to (HL) in ASCII
CHAN	24	18	Execute EMT given in reg C
SAVE	35	23	Save regs HL, DE and BC
SAVEA	36	24	As SAVE, also save AF
FSTLST	37	25	Input first and last 16-bit numbers
GETJP	38	26	Return Ath entry from table at (HL)
BASE	39	27	Return scratchpad addr in reg DE
VERSN	51	33	Return version number of COS

WAIT Wait B 833 usec intervals

The wait routine (.EDGE in the Monitor listing) does not return to the calling program until the number of 1/1200 second intervals specified by the contents of register B has elapsed. If B contains 0, the delay is 256 such intervals.

Registers affected: None

UPDATE Copy MASK to PORT0

A number of functions on the 380Z are controlled by the bit pattern written into RAM location PORT0 (see the Monitor listing for the exact address). This location is write-only, and when read in fact returns the current character available from the keyboard. It is therefore necessary to maintain a record of the current state of the bits in PORT0 and this is done at location MASK in the monitor scratchpad - see Monitor listing for the exact address. The normal procedure for updating PORT0 is to modify the mask word at location MASK and then copy this to PORT0. The UPDATE routine does precisely this; the contents of the mask at MASK are written to PORT0.

Registers affected: None

GETHEX Get hex number from keyboard into HL

GETHEX is the common input routine used by the Monitor and Front Panel commands to obtain a hexadecimal value from the user via the keyboard. It is the routine called for example after typing J, to obtain the address to be jumped to. GETHEX allows up to four hexadecimal digits to be input, each hexadecimal digit being a digit (0-9) or a letter in the range A-F (or a-f). Note that GETHEX allows input in free form so that error correction is possible by using the DELT key. The entered hexadecimal number is terminated by any non-hex character. Leading 0's are automatically inserted; if no

 Copyright October 1980

hexadecimal digits are entered, 0 is returned.

On return, register HL contains the entered hexadecimal value converted to binary with the more significant byte in H and the less significant byte in L. Register C contains the number of hex characters entered (0 to 4). Register B contains the terminating character in ASCII. Register A is undefined.

Registers affected: A,B,C,HL, and the flags

DEOUT Put register DE to (HL) in ASCII

The DEOUT routine is used to convert a 16 bit binary value, in register pair DE, to an ASCII string at the address contained in register pair HL. On entry HL can point to any valid memory address. 4 hexadecimal characters are output to this address and the 3 following locations. On return HL points to the location following that of the last character output. Leading zeros are inserted if appropriate. The contents of DE are unchanged.

DEOUT is used by the monitor primarily for writing to the screen but it can be used on any occasion where binary to hexadecimal ASCII conversion is required.

Registers affected: Register A and the flag register are destroyed, HL is incremented as described above.

BYTEO Put register A to (HL) in ASCII

Like DEOUT, BYTEO converts a binary value to a hexadecimal ASCII string, this time converting the 8 bit binary value in register A. ASCII characters are stored at the address given in register HL and at the following location, and on return HL points to the location following that of the last character stored. A leading zero is inserted if appropriate.

Registers affected: Register A and the flag register destroyed, register HL incremented as described above.

CHAN Execute EMT given in register C

This EMT call is provided so that a trap call can be made with a trap code which is not known at the time the program is written. To allow such programs to be stored in read only memory, the CHAN trap call is provided. This call works just like the normal trap call but the channel to be used is put in register C rather than in the trap code.

Registers affected: Depends on channel

 Copyright October 1980

SAVE Save registers HL, DE, and BC

SAVE is a routine which pushes registers HL, DE, and BC on to the stack, together with the address of a routine to pop them off again. For example, both of the following program fragments cause the values of these three registers to be preserved across the call to the routine SUBR.

```
SUBR:   EMT     SAVE
        .
        .
        .
        RET


SUBR:   PUSH    HL
        PUSH    DE
        PUSH    BC
        .
        .
        .
        POP     BC
        POP     DE
        POP     HL
        RET
```

One drawback of SAVE is that, although the value contained in HL is preserved across the called routine (SUBR in the examples), the old value is not preserved across SAVE itself, which means that the old value of HL is not available to the called routine.

Registers affected: HL

SAVEA As SAVE, also save AF

SAVEA behaves in much the same way as SAVE, except that register A and the flags are also pushed on to the stack in addition to BC, DE, and HL. Like SAVE, the value of HL is not available to the called routine, although it is preserved across the called routine.

Registers affected: HL

FSTLST Input first and last 16-bit numbers

FSTLST is a general purpose routine which obtains two 16-bit hexadecimal numbers from the keyboard. It first prompts

```
First>
```

and calls the routine GETHEX (q.v.) to obtain the first number. It then prompts

```
Last>
```

and obtains a second hexadecimal number, again using GETHEX. On exit, the first number is held in register pair DE and the last is in BC. HL contains the difference between these two quantities. Register A and the flags are destroyed.

If the first number is greater than the second, COS prints

```
?Err?
```

and control returns to COS command level.

Registers affected: HL, DE, BC, A, and the flags.

GETJP Return Ath entry from table at (HL)

GETJP is used to return an entry in a table of two byte quantities, often addresses. On entry, HL holds the address of the zeroth entry in the table, and A holds the entry number. On exit, HL holds the Ath entry in the table. For example, the following program fragment performs a 'computed GOTO'

```
          .
          .
          .
          LD      HL,TABLE        ;TABLE OF POINTERS
          EMT     GETJP           ;GET ENTRY
          JP      (HL)            ;GO TO ROUTINE

TABLE:    DEFW    SUB1
          DEFW    SUB2
          DEFW    SUB3
          .
          .
          .
```

Registers affected: HL

BASE Return scratchpad address in register DE

BASE returns in DE the address of the base of monitor scratchpad RAM. It was originally provided to cope with versions of the 380Z where the scratchpad address varied with memory size. The design was changed to render this unnecessary, but the EMT is still included to maintain compatibility.

Registers affected: DE

 Copyright October 1980

VERSN Return version number of COS

VERSN returns the version number of COS, multiplied by 10 to convert it to an integer, in register A.

Registers affected: A, HL, DE, BC, and the flags

PRINTER SUPPORT EMTS

EMT LPOUT is provided to output characters and control characters to a printer selected by EMT SETLST. EMT LPSTAT is additionally available for software to check that the selected printer is ready.

Mnemonic	Code	Hex	Function
LPOUT	5	5	Output byte in reg A to selected printer
SETLST	41	29	Set printer from regs A and E
S4KTL	47	2F	Test the SIO-4 for a character
S4KIN	48	30	Read SIO-4 into reg A
LPSTAT	50	32	Check that printer is ready

LPOUT Output character to selected printer

The character held in register A is sent to the currently selected printer device. For details of selecting a printer, see the description of the line printer handlers below under EMT SETLST.

This EMT is called via a transfer vector - see the monitor listing for the exact address. This means that alternative handlers can be installed, simply by adjusting the transfer to point to the new routine.

Registers affected: None

SETLST Set printer from registers A and E

SETLST sets up a printer device, connecting EMTs LPOUT and LPSTAT (qq.v.) to the appropriate printer handlers, and performing any necessary initialization. On entry, register A holds the printer type code, and E holds the Baud code, where appropriate.

The default setting for EMT SETLST is that all output via EMT LPOUT will be sent to the screen (similarly as with EMT OUTC) if a printer has not previously been selected. Pressing the reset button will redirect LPOUT back to the screen.

Available printer types are:

Code	Printer type
0	VT screen
1	SIO-1
2	SIO-2 or SIO-2B
3	Parallel (Centronics or Anadex DP8000 and DP9501)
4	SIO-4 (Qume Sprint 5)
5	SIO-5
6	SIO-6

A Research Machines Microline 80 may be fitted with either a parallel or serial interface. For more details of printer types, see Printer Support in Chapter 2.

Baud codes, where relevant, indicate the Baud rates shown in the following table:

Code	Baud rate
0	110
1	300
2	600
3	1200
4	2400
5	4800
6	9600

Printer type 0 is the screen, and all output directed to the line printer transfer vector via EMT LPOUT appears on the screen. This printer type does not require a Baud code.

Printer type 1 is used with SIO-1 and SIO-1B serial interfaces, which are identical from a software viewpoint. The Baud rate for this type of interface is selected by means of switches on the interface card itself, so no Baud code is required.

Printer type 2 is used with SIO-2 and SIO-2B serial interfaces. These require a Baud code, but, because these devices will not operate beyond 2400 Baud, the maximum Baud code allowable is code 4.

Printer type 3 is used for printers with a parallel interface, such as most Centronics and Anadex models. Being parallel, Baud rate has no meaning, and printer type 3 therefore requires no Baud code. If an error condition exists, COS will type

Attend to printer!

Possible causes are:

Printer not switched on.
Printer plugged into the wrong socket on the 380Z.
Wrong type of connecting cable.
Printer out of paper.
Printer not selected by select/on line switch on printer.

Printer type 4 is used with an SIO-4 serial interface. This type of interface will handle any Baud rate up to 9600, so allowable Baud codes range from 0 to 6.

Printer type 5 is a 'second' serial interface, an SIO-5. Allowable Baud codes are 0 to 6.

Printer type 6 is a third serial interface, an SIO-6. This is a special card, not normally fitted. Again, Baud codes in the range 0 to 6 are allowable.

The only difference between printer types 4, 5, and 6 is the I/O ports at which they are located. These are:

Suffix	FDS F	MDS M	Cassette C
Type 4	E8	C8	C8
Type 5	C8	E8	E8
Type 6	48	48	48

Registers affected: None

S4KTL Test the SIO-4 for a character

S4KTL checks the SIO-4 serial interface, and returns -1 (FF hex) in register A, with the zero flag reset, if a character is available, else 0 in A and the zero flag set. The interface must have been initialized previously, possibly by calling EMT SETLST with 4 in register A and an appropriate Baud code in register E.

Registers affected: A and the flags

S4KIN Read the SIO-4 into register A

S4KIN reads the SIO-4 serial interface, and returns the first available character in register A, stripping off the most significant (parity) bit. This routine does not return to the user program until a character is available, so to prevent a possible 'locking up' of the program a call to EMT S4KTL should be made first to ensure that a character is waiting. S4KIN assumes that the interface has been previously initialized, for example by an appropriate call to EMT SETLST.

Registers affected: A and the flags

LPSTAT Check that printer is ready

LPSTAT checks the printer device set up by EMT SETLST, and returns -1 (FF hex) in register A if the device is ready to receive a character, else 0 in A with the zero flag set. This routine can be used to prevent a call to EMT LPOUT 'locking

up', by checking beforehand that the printer is ready.

This EMT is called via a transfer vector - see the monitor listing for the exact address. This means that alternative handlers can be installed, simply by adjusting the transfer to point to the new routine.

Registers affected: A and the flags

DISC EMTS

Most users will access the FDS or MDS system via CP/M; for completeness, however, details of the disc support provided in the COS ROM are given here. CP/M accesses these routines via its 'RBIOS' section. Different driving routines are required for the MDS and FDS; the version present is indicated by the suffix /M or /F to the COS title printed after power-on or reset. However, all the disc EMTs except BOOT, go via transfer vectors, allowing switching, for example, between the /F and /M types.

Mnemonic	Code	Hex	Function
INIT	25	19	Initialise disc unit
RDSEC	26	1A	Read sector
WRSEC	27	1B	Write sector
WRCHK	28	1C	Write and check sector
BOOT	49	31	Perform a cold bootstrap

INIT Initialise disc unit

INIT initializes the floppy disc drive whose unit number is held in the memory location whose address is in IX. On return, an error code is returned in register A. Each bit represents a separate error, and when set means that the error has occurred. The meanings of the bits are:

Bit 7 = Not ready
Bit 4 = No track 00 signal

If the call to INIT was successful, on exit register A will be zero and the zero flag will be set.

This EMT is called via a transfer vector - see the Monitor listing for the exact address. This means that alternative handlers can be installed, simply by adjusting the transfer to point to the new routine.

Registers affected: A and the flags

RDSEC Read sector

RDSEC reads a sector from disc. The parameters to RDSEC are passed in a disc control block, whose address is in IX. The information in the control block is

```
IX -> DEFB    UNIT
      DEFB    TRACK
      DEFB    SECTOR
      DEFW    ADDRESS
```

UNIT specifies the logical unit number of the drive to be accessed. This should be a byte in the range 0-3, specifying units A, B, C, or D, respectively. On MDS systems, where there may be fewer than four logical disc drives, the allowable range may be less.

TRACK is the number of the track on which the sector to be read is located. On an FDS system, this can range from 0-76 (0-4C hex), while on an MDS the range is 0-39 (0-27 hex).

SECTOR is the sector number, which can range from 1-26 (1-1A hex) on an FDS or 1-16 (1-10 hex) on an MDS.

ADDRESS, which is a two byte quantity, specifies the memory address at which the data from the sector will be stored. This should of course be somewhere in read/write memory, although no check is made on this.

On exit, register A contains any error codes. As with INIT, each bit represents a different error, the errors being:

```
Bit 7  =  Not ready
Bit 4  =  Record not found
Bit 3  =  CRC error
Bit 2  =  Data lost
```

On exit, if no error has occurred, register A will contain zero and the zero flag will be set.

This EMT is called via a transfer vector - see the Monitor listing for the exact address. This means that alternative handlers can be installed, simply by adjusting the transfer to point to the new routine.

Registers affected: A and the flags

WRSEC Write sector, no check

WRSEC writes a specified sector on disc. The parameters are the same as for RDSEC (see above), except that ADDRESS on entry is the address of the data to be written out. On exit, register A holds the error codes, as with INIT and RDSEC. The error codes are:

Bit 7 = Not ready
Bit 6 = Write protect
Bit 5 = Write fault
Bit 4 = Record not found
Bit 3 = CRC error
Bit 2 = Data lost

On exit, if no error has occurred, register A holds zero and the zero flag is set.

This EMT is called via a transfer vector - see the Monitor listing for the exact address. This means that alternative handlers can be installed, simply by adjusting the transfer to point to the new routine.

Registers affected: A and the flags

WRCHK Write and check sector

WRCHK writes out the specified sector, exactly as for WRSEC, and then reads it back to verify the operation. Error codes, returned as usual in A, are:

Bit 7 = Not ready
Bit 6 = Write protect
Bit 5 = Write fault
Bit 4 = Record not found
Bit 3 = CRC error
Bit 2 = Data lost
Bit 1 = Data mismatch

On exit, if no error has occurred, register A holds zero and the zero flag is set.

This EMT is called via a transfer vector - see the Monitor listing for the exact address. This means that alternative handlers can be installed, simply by adjusting the transfer to point to the new routine.

Registers affected: A and the flags

BOOT Perform a cold bootstrap

This EMT causes a cold bootstrap of the disc system. Obviously, it does not work on cassette systems, instead returning control to COS command level. COS looks for the bootstrap on drive A or B, depending on whether the last cold boot performed 'manually' was achieved by the B or X command from COS command level. This EMT will not work if no cold bootstrap has been performed since power-on. If no disc is present on the selected drive, or there is some other error, COS will print

Disc not ready

 Copyright October 1980

and control returns to CP/M command level.

Registers affected: BOOT does not return to the calling program

CASSETTE EMTS

EMTs are provided in the cassette environment which read and write bytes of data on cassette and also set the cassette transfer speed. Disc versions of COS later than 3.2 do not support slow cassette speeds, instead a separate utility program SCASS must be used, described in Appendix C.

Mnemonic	Code	Hex	Function
PUTBYT	3	3	Output byte in reg A to tape
GETBYT	4	4	Read byte from tape into reg A
GETSYN	17	11	Get syn char in reg A from tape
SETCAS	40	28	Set tape speed from reg A

PUTBYT Output byte in register A to tape

The character in the A register is output to tape. Note that the character is output immediately, and that no file structure is imposed. Without file structuring, it will normally be difficult to find any given character on a tape.

On cassette systems, and on disc systems running versions of COS earlier than 3.3, cassette output may be at either 300 Baud or 1200 Baud. See the description of the cassette interface for details.

This EMT is called via a transfer vector - see the Monitor listing for the exact address. This means that alternative handlers can be installed, simply by adjusting the transfer to point to the new routine.

Registers affected: None

GETBYT Read byte from tape into register A

The next character read from tape is returned in register A. GETBYT does not return until either a character has been read from tape or a CTRL character is typed at the keyboard. In the former case, the character is returned in register A and the carry flag is cleared. Otherwise, the carry flag is set and A is undefined.

Like PUTBYT, GETBYT will read the tape at either 300 or 1200 Baud on cassette systems or on disc systems with versions of COS earlier than 3.3. See the description of the cassette interface for details.

This EMT is called via a transfer vector - see the Monitor

listing for the exact address. This means that alternative handlers can be installed, simply by adjusting the transfer to point to the new routine.

Registers affected: A and the flags

GETSYN Get synchronisation character from tape

This routine is used to find a synchronising character which marks the beginning of a block on cassette tape. The sync character to be searched for is put in register A. GETSYN first searches for an interrecord gap which is defined as 240 cycles of 2400 Hz tone, i.e. a 100 millisecond period on the tape without any recorded information. It then compares the next character on the tape with the sync character in A. If there is a match, GETSYN returns to the calling program, else it resumes looking for an interrecord gap. In this way the beginning of a block of information on tape is very closely defined and the tape can be started from rest while reading it without detecting erroneous characters. The COS and Cassette File System formats use the code 16 hex as the synchronisation character.

The keyboard is constantly monitored during GETSYN, and if any CTRL character (a character whose ASCII value is less than 1B hex) is typed, GETSYN is aborted, and returns to the calling program with the carry flag set. Otherwise, the carry flag is reset when GETSYN exits.

The detection of the interrecord gap is carried out by subroutine .GTGAP which is called via a Monitor transfer vector - see the Monitor listing for the exact address. This was done so that the basic Monitor load and dump routines could be used with paper tape instead of cassette tape. In the case of paper tape, an interrecord gap would be identified by different means, perhaps by looking for a number of null characters. The GETGAP transfer vector would need to be modified to point to a routine to do this.

Registers affected: None

SETCAS Set tape speed from register A

SETCAS sets the cassette speed between fast (1200 Baud) and slow (300 baud) for input and output independently. Bit 0 (the least significant) controls the speed of input - set for fast, reset for slow. Bit 1 similarly controls output speed. See the description of the cassette interface for further details.

On entry, bits 0 and 1 of register A should hold the desired new speed settings. On exit, A holds the old speed settings in bits 0 and 1, with the other six bits being reset.

Disc versions of COS later than 3.2 do not support slow

cassette I/O. With these versions therefore, the value of A on entry is ignored, and A is always set to 3 on exit. In order to utilise slow I/O, such users should use the SCASS utility program described in Appendix C.

Registers affected: A and the flags

 Copyright October 1980

I/O TRANSFER

I/O CHANNELS

Transfer of data between external devices (e.g. the keyboard) and the CPU is organised by COS around the concept of 'channels', an Input - Output (I/O) channel being defined as a fixed point of reference through which all bytes of a particular data stream have to pass. These fixed points are the 'transfer vectors'. Linkage from a user program to the desired transfer vector is provided by a selection of trap calls. The first ten EMTs access the major I/O channels, with the trap code representing the channel number. Linkage to the target subroutines which do the actual transfers is provided by address information stored in the transfer vectors; these are initially set up by COS but, being in random access memory, can be changed if required. Return to the calling program takes place in the usual way, the return address having been pushed onto the stack when the initial call was made, so that a 'RET' instruction at the end of the peripheral handling subroutine will return control correctly.

COS provides a total of 10 I/O channels; channels 1 - 5 are assigned by convention to the screen, the keyboard, cassette output, cassette input, and the line printer (any printing device) respectively, whilst channels 6 to 10 are for data transfers to and from cassette files and are unallocated on disc systems. Together with the EMTs 1 to 10 decimal, there are also an additional seven EMTs which pass through transfer vectors:

Mnemonic	Code	Hex	Function
OUTC	1	1	Output byte in reg A to screen
KBDC	2	2	Read keyboard, trap CTRL C
PUTBYT	3	3	Output byte in reg A to tape
GETBYT	4	4	Read byte from tape into reg A
LPOUT	5	5	Output byte in reg A to selected printer
OUT1	6	6	EMT via transfer vector (spare)
OUT2	7	7	As OUT1
IN1	8	8	EMT via transfer vector (spare)
IN2	9	9	As IN1
IN3	10	A	As IN2
INIT	25	19	Initialise disc unit
RDSEC	26	1A	Read sector from disc
WRSEC	27	1B	Write sector to disc
WRCHK	28	1C	Write and check sector
KBDTL	31	1F	Test keyboard, return -1 (true)
KBDWF	34	22	Wait for key struck, trap CTRL F
LPSTAT	50	32	Check that printer is ready

The cassette channels (3 and 4) allow direct I/O to a cassette recorder on a single byte basis but are not necessarily limited to this medium. They are merely assigned to the function of data storage; thus, if suitable device handlers were provided, they could be used to interface to a paper tape punch and reader in a paper tape based system. The COS monitor contains device handlers for channels 1 to 5; the behaviour of these has been described in detail earlier in this section.

Channels 6 to 10 are intended for I/O to files and are initialised and used by the Cassette File System (CFS). Thus programs, such as the Text Editor, which can run with CFS and require access to data files, transfer data to and from such files by means of trap calls with codes in the range 6 to 10. These are then processed by CFS (which may itself execute a trap call with code 3 if the data transfer required a data block to be written on tape).

I/O TRANSFER VECTORS

Input and output (I/O) of data bytes, as described above, is normally done via the transfer vectors. These are 3 byte areas of random access memory (RAM), which are set up by the monitor when power is first applied or after pressing the Reset button, to contain a 3 byte jump instruction (C3 nn hex). The addresses of the transfer vectors can be found both in the Monitor listing and in Appendix A. They work as follows:

As an example, consider a 'read' operation. The calling routine calls the transfer vector, normally by means of an EMT call on an appropriate channel. The transfer vector contains a jump to the input subroutine proper. This routine obtains a character from the input stream, and returns to the calling level via a RET instruction.

Using transfer vectors for I/O in this way has two advantages. The calling program does not need to know the location of the I/O subroutine, only the transfer vector code. Thus programs can run with different versions of COS as long as the transfer vectors are present. Also the actual I/O subroutine used can be changed, by changing the transfer vector, without changing the program. A common example of this in RML software is line printer output to transfer vector 5; when there is no line printer in the system, this transfer vector points to the output routine to the screen but when a line printer is present, it is changed to point to the appropriate subroutine. Thus the general concept of transfer vectors allows programs such as BASIC to perform I/O transfers in a device independent manner.

NON - I/O TRANSFER VECTORS

In addition to the I/O Transfer Vectors described in the previous section several other transfer vectors are defined by COS. These will be described now:

GETGAP

This vector is initialised by COS to the address of label .GTGAP and serves to link the .GTSYN routine to the .GTGAP subroutine. The first part of the process of finding the beginning of a data block in a cassette file consists of finding an inter-record gap, and this is what .GTGAP does. However .GTSYN calls .GTGAP via the GETGAP transfer vector so that it can be skipped where appropriate (e.g.when the file is on paper tape), by replacing the jump instruction (C3 hex) at label GETGAP with a return instruction (C9 hex). .GTSYN is available via a Trap Call, q.v.

MONX

This vector is initialised to the RET instruction (C9 hex) and therefore returns immediately when called from the monitor main command loop, which is entered whenever CTRL C is typed (label ZMON). However, after initialisation, but before the main loop is entered for the first time, COS checks for additional ROM at address F600; if the content of location F600 is zero, COS calls it. These features are intended to allow COS to be extended. The code at address F600 would place a jump instruction at MONX and the address of the extended monitor entry point at MONX + 1, then return. COS would then enter the extended monitor immediately after typing its arrow prompt.

However as long as the contents of location F600 are non-zero (which is also the default case with no additional ROM plugged in), this area of memory is available for other purposes and COS will ignore it.

The address of MONX is given in Appendix A.

TRAPX

Control passes to this vector when a trap call with a code outside the permitted range is executed. If it is intended to use such codes, a jump instruction (C3 nn hex) should be stored at TRAPX. Note that at this point the stack looks like this:

```
High
        Old PC

        Old HL

        Old AF
Low             SP
```

Register A contains the trap code and Register HL contains the address at which the trap code was stored. 'Old PC' is the return address, assuming normal EMT conventions, i.e. the address of the location following the one containing the trap code. 'Old AF' and 'Old HL' are the contents of these registers at the time the trap call was issued. These must be popped before a 'RET' instruction can occur.

If an EMT is executed during normal operation which has a code outside the permitted range, and which has not been set up, a 'Break' will be produced at TRAPX, the address of which can be found in Appendix A.

 Copyright October 1980

PERIPHERAL INTERFACING

USER I/O PORT

The standard 380Z includes an 8-bit parallel interface referred to as the user I/O port. This port is 'memory-mapped' to address FBFF hex; writing a data byte to this address (e.g. by the instruction 'LD (0FBFFH),A') causes the data to be latched onto the output lines.

The port can be used, for example, for driving a serial device such as a teletype, for a parallel device such as a printer, or for other purposes.

The connector for the user I/O port is at the top left of the CPU board looking at the component side, i.e. the connector nearest the rear of the cabinet with the CPU board in its normal position. Pin 1 is marked on the connector. The diagram below shows the pin assignments, viewed from above the male connector on the CPU board. The inputs and outputs are from standard low-power Schottky TTL Devices, viz.74LS244 for input, 74LS374 for output. Refer to data sheets for further details. Only low currents should be drawn from the power pins.

Example programs for driving a 20mA or RS232 serial device via the RML SIO-2 and SIO-2B interfaces connected to the user I/O port can be found in the Option ROM listing. Note that the SIO-2 is necessary to condition the signals to the appropriate standard.

Pin 1	DI0	. .	DI1	Pin 2	
Pin 3	DI2	. .	DI3	Pin 4	
Pin 5	DI4	. .	DI5	Pin 6	
Pin 7	DI6	. .	DI7	Pin 8	DIn = Data in
Pin 9	DO0	. .	DO1	Pin 10	
Pin 11	DO2	. .	DO3	Pin 12	DOn = Data out
Pin 13	DO4	. .	DO5	Pin 14	
Pin 15	DO6	. .	DO7	Pin 16	
Pin 17	+5V	. .	-12V	Pin 18	
Pin 19	+12V	. .	0V	Pin 20	

KEYBOARD

The keyboard interface consists of an 8-bit latch in which data from the keyboard appears after a key depression. The latch is memory-mapped to address FBFC hex. Thus the instruction

```
LD A,(0FBFCH)
```

transfers the ASCII code of the most recently pressed key to the A register. Keyboard data is entered into the keyboard latch by a strobe pulse, generated by the keyboard each time a key is depressed.

So that programs (including COS) can detect when a key has been depressed, RML's convention is that whenever the keyboard latch (referred to from now on simply as the keyboard) has been read, it is cleared.

 Copyright October 1980

Thus, when a zero byte is read (ASCII null, not generated by the keyboard), no key has been struck since the previous one was read. This process is referred to as 'clearing the keyboard'.

Bit 0 of Port 0 at address FBFC clears the keyboard if set. Thus to clear the keyboard, bit 0 is set in the Port 0 Mask byte (contents of byte at lable Mask), the Mask byte is written to Port 0, bit 0 is cleared and the mask byte again written to Port 0. This is normally done using a trap call to UPDATE. The indirect procedure is necessary because the current state of Port 0 cannot be read directly (see also description of EMT UPDATE).

The Monitor code at label .KBDIN reads the keyboard, clearing it if necessary and returns the current character if any in the A register, else zero, with the Z flag set appropriately. This is normally called by a trap call to KBDIN. Note that since bit 7 is not connected on the RML keyboard, it is necessary to mask it off (by the instruction AND 7FH).

THE VIDEO DISPLAY

The RML Video Display is a direct display of a 960 character block of static memory arranged as 24 lines of 40 characters, on an unmodified domestic TV set or video monitor. A TV set is connected via its aerial socket, the signal provided being in the UHF range from an internal modulator. Both UHF and video outputs may be used concurrently, and up to six monitors or TVs may be used simultaneously (with reduced clarity). The number of characters per line was chosen as the largest which can be displayed clearly on a domestic TV set and is the same as that adopted for 'Teletext' and 'Viewdata'; the characters are of course somewhat sharper on a video monitor.

The memory from which the display is refreshed is at locations F000 to F5FF, and is accessible to both the video circuitry and the CPU, though not to both simultaneously. When the video circuitry has control, a picture is displayed and CPU access to these locations has no effect - a read operation returning FF hex, whilst, when the CPU has control, the screen is blank. Access is decided by bit 2 of the memory mapped port PORT0, whose exact address is given in the monitor listing. Writing a 0 bit into this position grants access to the video circuitry, a 1 bit to the CPU.

It can be seen, therefore, that the behaviour of the screen is entirely dependent upon the software which drives it. It is thus rather different from conventional VDUs where the characteristics are set in hardware, and offers considerable advantages: split screen displays, extremely rapid update, and variable scrolling schemes are a few of these.

SCREEN TIMING

The computor can gain access to the video memory at any time by setting the control bit in PORT0, as mentioned above. However, if this occurs whilst the visible portion of the display is being refreshed, a disturbance of the picture will be noticeable, consisting of either a momentary dark horizontal band with height depending on how long the CPU

has control of the video memory, or a brief bright flash. Means are therefore provided to synchronise the driving program to the video circuitry such that the CPU only takes control during the frame and line blanking periods. Two bits in PORT1 (See Appendix A for the exact address) indicate the occurrence of these periods. Bit 6 becomes high during frame blanking, that is, for about 4.5 milliseconds every 20 milliseconds. This period is normally used when several characters have to be transferred, or during scrolling. Bit 7 becomes high during line blanking. The relationship of this level to line blanking has been adjusted such that there is time to output a single character - see the Monitor listing at address .OUTF for an example. This allows high speed output of text without visible disturbance, and is used for example in the scroller to give an overall transfer rate (including scrolling) equivalent to around 5000 Baud.

FRAME OUTPUT

Because of these timing restrictions it is a little simpler to output to the screen during the frame blanking period, and the COS monitor provides some support routines to simplify this still further. As described in the section dealing with the Monitor Trap Calls, the routine OPNWT (really 'wait', then 'open' screen memory to CPU) returns during the next frame blanking period with the screen memory open; conversely the routine CLOSE closes it again (immediately). Thus, providing the instructions take less than 4.5 milliseconds to execute, code to manipulate the screen memory as desired can be placed between calls to OPNWT and CLOSE, without disturbing the displayed picture in any way. The following simple example would cause the letter 'A' to be displayed at the top left corner of the screen:

```
Code        Assembler Syntax

21 00 F0    LD      HL,0F000H
F7 0B       EMT     OPNWT
36 41       LD      (HL),'A'
F7 0C       EMT     CLOSE
```

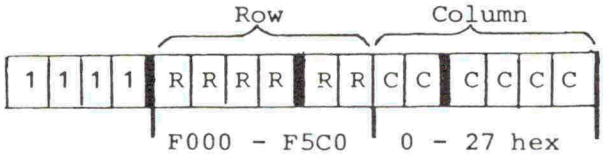
Many more complex examples can be found in the COS Monitor listing, including the code of the Monitor Scroller (label .OUTC) and the mechanism of OPNWT and CLOSE.

LINE OUTPUT

As mentioned above, the timing constraints for accessing the screen memory during the line blanking period are rather severe, even with the Z80A. The subroutines .OUTF and .INF, in the Monitor listing, which respectively output the character in register A to the display address given in register pair HL, and input the character at the address given in HL into A, should serve as a guide. These routines are used for most single-character access to the screen within COS, most notably in the scroller.

SCREEN ADDRESSING

The addresses of the various character positions within the display memory are 16 bit numbers, covering the range F000 to F5E7 hex, and are made up as follows:



Thus the addresses of the first row run from F000 to F027 hex (giving 40 decimal character positions). However the various rows are not contiguous in address space; the second row runs from address F040 to F067, the third from F080 to F0A7 and the last from F5C0 to F5E7. This can easily be verified from the bit map given above, when it will be seen that the increment in address between the first character position of each row (i.e. the increment down a column) is 40 hex and that there are 24 (decimal) rows in all. If you intend programming the screen memory directly, it is a worthwhile exercise to make yourself a 'map' of the screen with the addresses of useful points on it.

It can be seen that there are several 'missing' addresses (e.g. from F028 to F03F). Writing (or reading) to these does map onto the screen, but in a fairly unpredictable way and is best avoided.

N.B. You can experiment with the memory using the Front Panel commands (which automatically open and close the display memory to allow this).

CASSETTE INTERFACE

The cassette recorder interface comprises a tone generator with shaping circuits which can output a near sinusoidal wave form at 2400 Hz or 1200 Hz, input shaping and filtering circuits to allow the frequency and amplitude of such tones to be measured when replayed, and two output lines which can be used for motor control.

OUTPUT TO CASSETTE

The output signal at 2400 Hz or 1200 Hz is at low level, suitable for the 'microphone' input of most cassette recorders. To ensure a clean signal, a 47 nF capacitor should be connected between the signal line and ground at the recorder end. A 2400 Hz signal is present when bit 4 of Port 0 (at FBFC hex) is set, 1200 Hz when it is cleared. COS sets this bit when power is first applied or after a Reset, so that a signal at 2400 Hz is present when COS is quiescent. When data is being output to tape the signal frequency is changed appropriately. In order that frequency transitions should occur cleanly, means is provided to synchronise them to the moment of base-line crossing of the output signal. The 1200 Hz signal can be read in bit 1 of Port 1 (at FBFE hex). Synchronising the transition of frequency to the edge of the latter ensures that frequency transitions of the output signal always occur at the same point on the

wave form, and that bursts of 2400 Hz consist of a multiple of 2 such cycles.

COS supports the writing of data bytes to the 'Kansas City' standard (also called the CUTS standard). An 8-bit data byte is serialised, the least significant bit being sent first, preceded by a start bit and followed by at least one stop bit. The resetting state is 2400 Hz (the '1' state) and each bit consists of 8 cycles of 2400 Hz or 4 cycles of 1200 Hz for the '1' bit and '0' bits respectively. The start bit is a '0' bit, the stop bits '1' bits. Thus the stop bits represent a return to the resting state, the next start bit being indicated by a 2400 Hz to 1200 Hz frequency shift.

COS subroutine .PTBYT carries out this serialising process and is normally reached by a trap call, code PUTBYT. The byte in register A is written to tape with one start bit and two stop bits at a rate of 300 bits per second. This gives an overall data rate of 27.3 bytes/sec. Subroutine .PTBIT outputs each bit, determining whether a frequency transition is required, and if so, looking after the timing along the lines described above.

INPUT FROM CASSETTE

The signal from the cassette recorder on replay should be at high level (around 1.5 V RMS)and is normally taken from the recorder DIN connector or external loudspeaker socket. This is shaped and filtered and can then be read on 2 bits of Port 1. Bit 5 is the replay signal and is used for decoding the tape data; bit 3 is the same signal, somewhat attenuated.

The replay signal is of adequate amplitude ('loud enough') when bit 3 shows occasional transitions from the resting '1' state to the '0' state.

COS decodes data bytes from tape by a rather complicated algorithm contained in subroutine .GTBYT. This is normally called by a trap call, code GETBYT and returns the byte read from tape in the A register. Basically the subroutine first searches for a start bit, then decodes the 8 ensuing data bits, by measuring the duration of each cycle replayed from tape. Subroutine GETCYC carries out the actual measurement and compares each duration with the constant VALUE, in order to decide whether the cycle was 2400 Hz or 1200 Hz. .GTBYT makes a majority decision as to whether each bit is a '0' or a '1' by counting the number of 1200 Hz and 2400 Hz cycles.

There is thus quite a considerable error margin and to a reasonable extent the code is self correcting. When a cassette recorder is properly set up, errors on replay are rare and can almost always be attributed to a defect of tape coating (a 'drop-out') or to external interference (e.g. from a refrigerator thermostat) reaching the cassette recorder via its mains lead.

There is one disadvantage of this method of tape decoding. The polarity of the signal on replay must be such that the frequency transitions occur on the rising slope of the sinusoidal signal, as it crosses the baseline. If it is inverted with respect to this convention, GETCYC will measure half a cycle of 2400 Hz and half a cycle of 2400 Hz and half of

 Copyright October 1980

1200 Hz, whenever there is a transition, with an even chance of identifying the cycle incorrectly. However the latter condition can easily be corrected in a number of ways and RML feel it is a small complication, compared to the increased reliability achieved by this method of tape decoding, relative to other methods in current use.

CASSETTE RECORDER MOTOR CONTROL

Bits 3 and 5 in Port 0 are intended to be used to control the drive motors of two cassette recorders and are available on the 7-way DIN Cassette Socket and the VDU board 26-way connector. (See Hardware notes for pin numbers).

By convention, bit 3 is used for Recorder 1, which is connected for reading (replay). Both recorders are connected for writing (record), but in two-recorder set up (supported by the Cassette File System and needed for the Text Editor, Assembler and BASIC with Data Files), bit 5, controlling Recorder 2 motor, controls writing. (Designating Recorder 1 for reading and Recorder 2 for writing keeps both the software and the actions required of the user simple!)

When bits 3 and 5 are set to '0', the recorder motors are energised; setting them to '1' stops the motors. In Monitor command mode (i.e. after CTRL C) the motors are on. This and the dual connection for writing mean that the simple one-recorder set up and the two-recorder one with motor control are compatible.

OTHER CASSETTE STANDARDS

Because the Cassette Interface is under software control, it can be programmed to operate according to standards other than 'Kansas City'. The Option ROM contains routines which interface with COS to provide cassette storage at 1200 Baud, i.e. a data rate 4 times faster than the Kansas City Standard. Bear in mind, however, that at this higher rate there is correspondingly less tolerance of errors due to tape drop-outs, external interference, etc.

Further information about cassette interfacing and operation is contained in the diagnostic and hardware manuals.

MISCELLANEOUS FUNCTIONS

RESET AND INITIALISATION

Mention is made throughout this manual of actions which occur when COS is initialised. These are summarised here:

When the Reset button is pressed, the Z80's reset line is pulsed, during which measures are made to allow dynamic memory to be refreshed. When the reset button is released, or when power is first applied, the initialisation code following label 'BEGIN' can then executed.

The initialisation code is not entered after CTRL C has been typed or an EMT 0 instruction has been executed; in these cases, control is transferred to label 'CONTC'.

Initialisation comprises the following phases:

1 Monitor work variables and transfer vectors OUT1 to IN3, NMIX and TRAPX are set to FF hex (the 'Break' code); transfer vector MONX is set to C9 hex ('RET'). The standard transfer vectors are set to jumps to their appropriate addresses.

2 The code following label TOPLP finds the address of the first unavailable RAM location in the contiguous block beginning at 100 hex and stores this at location 'HIMEM'. This is done without altering memory contents.

3 A test is made for monitor extension routines. If address F600 hex contains zero, a monitor extension is assumed and address F600 is 'called'.

4 COS checks to see if there is a VDU plugged into the SIO-4 socket. If there is, all subsequent output via the scroller output EMTs is sent to this VDU. Graphical output, such as the Front Panel, or low or high resolution graphics, still appear on the RML screen. If a character is received from the VDU at 9600 Baud, COS also switches the keyboard EMTs to take all subsequent input from the VDU ignoring the RML keyboard.

In addition, the keyboard and screen are cleared and COS types its title and version number.

POSITION INDEPENDENT CODE

It is often convenient to be able to write sections of code which can run at any address at which they happen to be loaded. Such code is referred to as Position Independent Code, or PIC. It is particularly useful in the cassette environment for subroutines which will frequently be reused, for it means that they can be attached to a variety of programs without having to be relocated.

The Z80 instruction set includes some, but not all, of the types of instructions needed for writing PIC. Essentially the requirements are

that instructions normally using an absolute 16-bit memory address should be replaced by ones using relative addresses. These include instructions to jump, call a subroutine and to load data from memory into a register and vice-versa.

The Z80 possesses a variety of relative jump instructions but these only have a range of -126 to +129. This generally presents little difficulty. It also possesses a number of useful 'register indirect' load instructions (e.g. LD A,(HL), LD (IY+3),B) which can be used in PIC to access memory if a means is provided to set up the registers used for addressing. The Z80 does not, however, have a relative call instruction, instead, COS provides this:

COS interprets a relative call instruction, for which the mnemonic CALR (code EF hex) has been defined, in much the same way as it interprets the trap instruction. In this case, the byte following the CALR is taken as the offset distance to branch, just as it is for the relative jump instructions (i.e. the offset is calculated as the relative distance from the byte following the offset). This can be used wherever a CALL would be used, provided the target subroutine is within range and bearing in mind that, being an interpreted instruction, it takes slightly longer to execute.

CALR can additionally be used in a PIC program to set up an absolute address in an index register, as the following example illustrates:

```
CALR    $+2
LD      DE,BASE-$    ;(SP) = Here
POP     IY
ADD     IY,DE        ;IY = BASE
```

where 'BASE' is the label of the desired target address, and $ is the symbol for the assembler's location counter.

This makes it possible to write quite complex programs entirely in PIC, such that they are not self-modifying and occupy little more space than a 'normal' program.

INTERRUPTS

Although the COS Monitor does not use Interrupts it can readily co-exist with routines which use them. COS is intended to be used with Interrupt Mode 2, the standard mode for Zilog peripheral devices. In this mode, the device supplies the lower 8 bits of an 'interrupt vector', the upper 8 bits having been previously set in the I register. Together these two halves form an address which should be set up prior to an interrupt occurring to contain the address of the appropriate interrupt service routine.

For the purposes of the following example, it will be assumed that the user wishes to service interrupts generated by the Counter Timer (Z80 CTC); the procedure for the other support chips is similar. The program servicing the interrupts should contain an 8 byte block which has been initialised to contain the addresses of the service routines for each of the four timer registers. For example, the assembly syntax:

```
INTTBL: DEFW ISR1
        DEFW ISR2
        DEFW ISR3
        DEFW ISR4
```

will generate such a table, assuming ISR1 to 4 are the labels of the service routines for the four possible interrupts. The following code would set up the CTC and the I register:

```
DI
LD      A,INTTBL/256
LD      I,A
LD      A,INTTBL AND 255
OUT     (CLK),A
IM2
EI
RET
```

where 'CLK' is the address of PORT 0 of the CTC on the I/O Bus.

Each CTC interrupt service routine (ISR1 to 4) should save and restore any registers it uses. Interrupts are automatically disabled when they occur, and must be re-enabled within the service routine. The routine should exit by a RETI instruction. Also see the appropriate Technical Manual for further details.

Certain precautions are necessary when using interrupts with COS. COS subroutines which involve timing can go wrong if interrupted; in particular, routines involving the VT screen, disc and tape I/O are not protected. Consult the Monitor listing if in doubt.

MEMORY ALLOCATION AND MANAGEMENT

Because of the various uses CP/M makes of the area of memory from address 0000 to 0100 hex, it has been necessary to reserve an area for the COS work area at the top of the address space. Currently COS uses the top 1K bytes, and as a result of this, the largest CP/M system you can run is 1K less than the memory size (e.g. 31K CP/M in a 32K system) except in the case of a 56K system.

To allow both user and system programs to be memory size independent, the memory pointer HIMEM, at locations 0006 and 0007 is used. In the cassette environment the contents initially contain the address of the top of memory plus one, but after CP/M has been loaded it changes to the lowest address used by the resident part of CP/M (the BDOS). It is important to note that the address in HIMEM is the first unavailable address of memory.

In a cassette system, a common example making use of HIMEM is the loading of a device handler for a printer. If this can be loaded, prior to BASIC for example, in such a way that it runs at the top of memory, with HIMEM being adjusted to point just below the device handler, BASIC need know nothing about it, yet all output to the printer (e.g. using LPRINT) will be output via this handler.

In a cassette system, the necessary steps are:

1 Move the handler to the top of memory, such that it occupies addresses from the current contents of HIMEM downwards.

2 Alter memory location HIMEM so that its content is an address just below the device handler.

3 Link the handler to the appropriate Transfer Vector.

In a disc system, altering HIMEM will not be sufficient to load the handler, as it will be written over by CP/M. The steps necessary are:

1 Reduce the size of CP/M using MOVCPM, and then bootstrap this new CP/M into the system.

2 Load the handler at the reserved memory space using DDT.

3 Link the handler to the appropriate Transfer Vector.

Note that the procedure is general; several device handlers may be resident at the top of memory at one time.

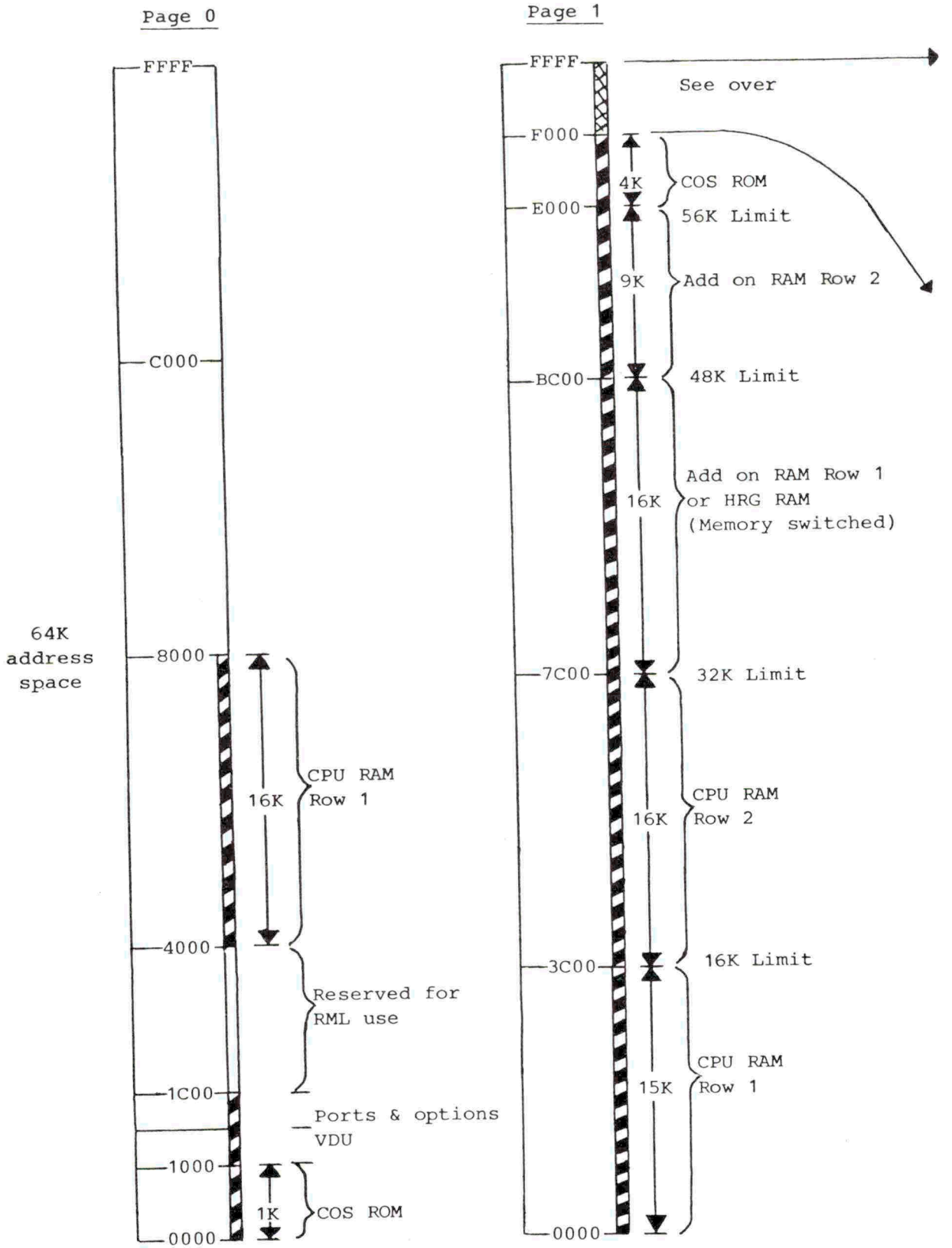
MEMORY MAP

The following pages illustrate the standard 380Z memory map and also the I/O port map. Page 0 of memory is not used in normal operation, it is only included here for completeness.

THIS PAGE INTENTIONALLY LEFT BLANK

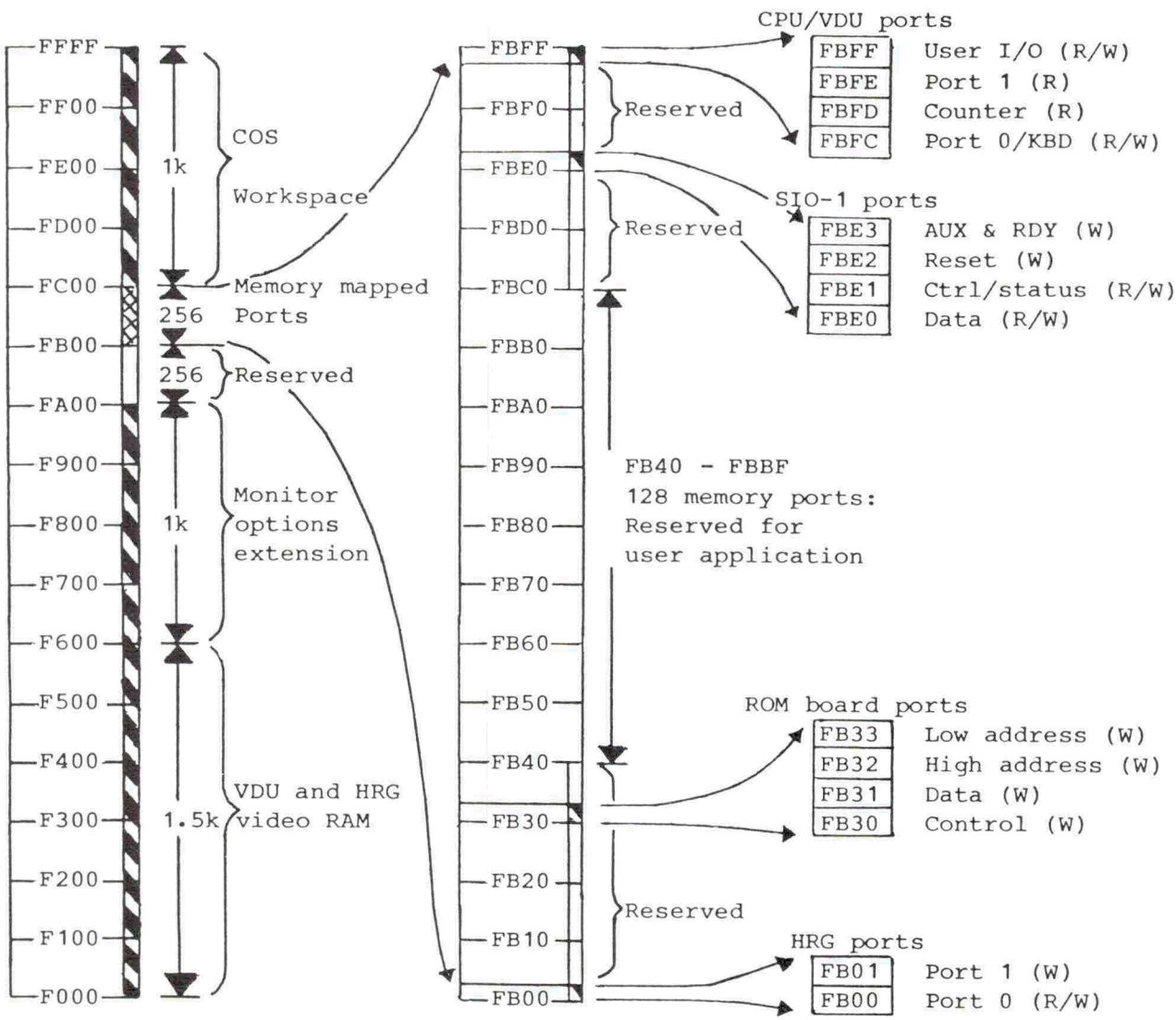
 Copyright October 1980

Standard 380Z Use of Memory



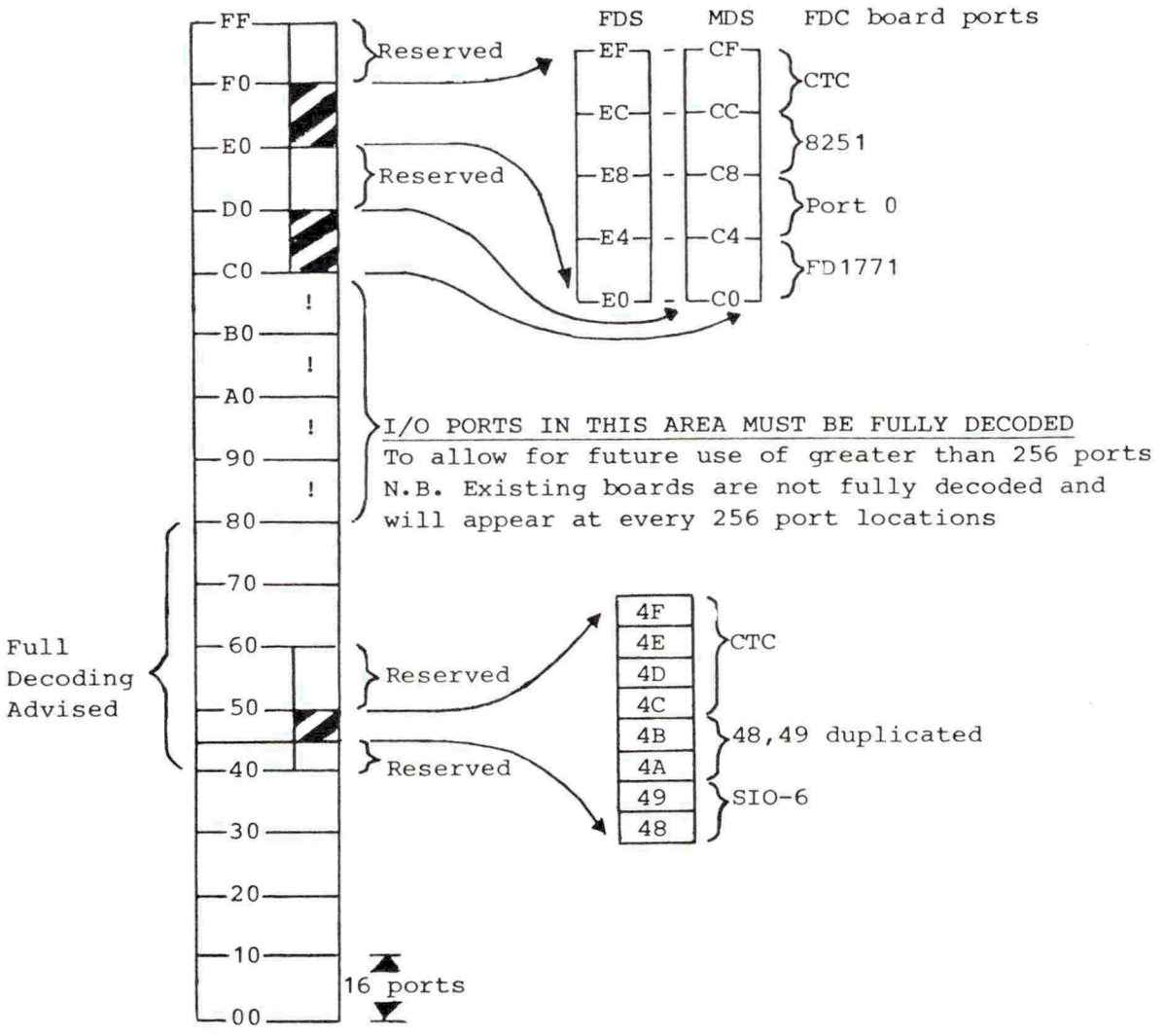
 Copyright October 1980

Page 1 Use of Memory above F000



 Copyright October 1980

Standard 380Z I/O Map



N.B. PIO boards occupy 16 consecutive I/O ports.
PIO + IEEE board occupies 32 consecutive I/O ports.
Users are advised to fill I/O ports from the bottom avoiding 40 to 5F if possible.

 Copyright October 1980

APPENDIX A

QUICK REFERENCE GUIDE

SUMMARY OF TRAP CALLS

Mnemonic	Code	Hex	Function
ERROR	-1	-1	Print ?Err?, return to Command Level
CONTC	0	0	Return to Command Level
OUTC	1	1	Output byte in reg A to screen
KBDC	2	2	Read keyboard, trap CTRL C
PUTBYT	3	3	Output byte in reg A to tape
GETBYT	4	4	Read byte from tape into reg A
LPOUT	5	5	Output byte in reg A to selected printer
OUT1	6	6	EMT via transfer vector (spare)
OUT2	7	7	As OUT1
IN1	8	8	EMT via transfer vector (spare)
IN2	9	9	As IN1
IN3	10	A	As IN1
OPNWT	11	B	Open screen in frame blanking
CLOSE	12	C	Close screen
*GRAFIX	13	D	Clear screen, 4 line scroller
*SCROLL	14	E	Full screen scroller
*CLEAR	15	F	Clear selected screen area
WAIT	16	10	Wait B 833 usec intervals
GETSYN	17	11	Get sync char in reg A from tape
UPDATE	18	12	Copy MASK to PORT0
GETHEX	19	13	Get hex number from kbd in reg HL
DEOUT	20	14	Put reg DE to (HL) in ASCII
BYTEO	21	15	Put reg A to (HL) in ASCII
*OUTC	22	16	As EMT 1 (not via transfer vector)
MSG	23	17	Output message at (HL) to screen
CHAN	24	18	Execute EMT given in reg C
INIT	25	19	Initialise disk unit
RDSEC	26	1A	Read sector
WRSEC	27	1B	Write sector
WRCHK	28	1C	Write and check sector
KBDIN	29	1D	Read and clear keyboard
KBDTC	30	1E	Test keyboard, return character
KBDTL	31	1F	Test keyboard, return -1 (true)
KBDTF	32	20	Test keyboard for CTRL F
KBDW	33	21	Wait for character from keyboard
KBDWF	34	22	As KBDW, trap CTRL F
SAVE	35	23	Save regs HL, DE and BC
SAVEA	36	24	As SAVE, also save AF
FSTLST	37	25	Access COS routine
GETJP	38	26	Return Ath entry from table at (HL)
BASE	39	27	Return scratchpad addr in reg DE
SETCAS	40	28	Set tape speed from reg A
SETLST	41	29	Set printer from regs A and E
OUTNC	42	2A	Scroller output without clearing page key
*OUTF	43	2B	Output to screen at (HL) from reg A
*INF	44	2C	Input from screen at (HL) to reg A
S4KTL	47	2F	As KBDTL, but from SIO-4

 Copyright October 1980

Mnemonic	Code	Hex	Function
S4KIN	48	30	Read SIO-4 into reg A
BOOT	49	31	Perform a cold bootstrap
LPSTAT	50	32	Check that printer is ready
VERSN	51	33	Return version number of COS

* These EMTs may not be supported in future versions of COS

SUMMARY OF OUTC CONTROL CHARACTERS

Character	Code	Hex	Function
CTRL D	4	4	Resume output (after CTRL O)
CTRL H (BACKSPACE)	8	8	Cursor left (non-destructive)
CTRL I (TAB)	9	9	Horizontal tab (non-destructive)
CTRL J (LINE FEED)	10	A	Cursor down
CTRL K	11	B	Cursor up
CTRL L (FORM FEED)	12	C	Clear screen, cursor to bottom left
CTRL M (RETURN)	13	D	Carriage return/linefeed
CTRL N	14	E	Carriage return
CTRL O	15	F	Suppress output
CTRL Q	17	11	Clear paging counter
CTRL S	19	13	Set paging counter
CTRL U	21	15	Blink on
CTRL V	22	16	Initiate XY addressing
CTRL W	23	17	Blink off
CTRL X	24	18	Cursor right (non-destructive)
CTRL Y	25	19	Clear to end of line
CTRL]	29	1D	Cursor home (top left)
CTRL ↑	30	1E	Clear to end of line
CTRL _	31	1F	Cursor home and clear screen
DELT (RUBOUT)	127	7F	Destructive backspace

SUMMARY OF MONITOR COMMANDS

B	Bootstrap CP/M from disc and execute
X	Bootstrap CP/M from disc drive B
L	Load program from cassette
D	Dump memory to cassette
C	Continue program at restart address
J	Jump to address and execute program
O	Select printer option
M	Enable High Resolution Graphics board as memory
CTRL C	Return to Command Level
CTRL F	Enter Front Panel mode
CTRL T	Enter Typewriter mode

 Copyright October 1980

SUMMARY OF FRONT PANEL COMMANDS

RETURN	Increment memory pointer
-	Decrement memory pointer
LINE FEED	Advance memory pointer by 8
/	Move memory pointer back by 8
CTRL L	Advance memory pointer by 32
CTRL O	Move memory pointer back by 32
M	Set memory pointer
I	Set memory pointer from memory contents (absolute)
R	Set memory pointer from memory contents (relative)
U	Set memory pointer from PC contents
.	Increment register pointer
X	Switch register display to alternate set
V	Update PC from memory pointer
G	Get first occurrence of specified pattern of bytes
N	Find next occurrence of pattern
H	Hexadecimal calculator
P	Fill and test memory between limits
S	Shift memory content
O	Set printer option
J	Jump to address and start execution
K	Continue program execution from PC
Z	Single step
<	Increment I/O port pointer
>	Decrement I/O port pointer
,	Write byte to I/O port
CTRL B	Return to monitor command level
CTRL C	Go to either command level or to CP/M if booted

LISTING OF COS WORKSPACE

This is an edited listing of the memory used by COS for the storage of transfer vectors, ports and main work variables. A full listing of the monitor is available on request.

```
FF00    NLINES: DEFS    1               ;Lines to scroll
FF01    TOP:    DEFS    2               ;Top of scrolled area
FF03    MASK:   DEFS    1               ;Port 0 Mask
FF04    VTPTR:  DEFS    2               ;VT Pointer
FF06    MPTR:   DEFS    2               ;Memory Pointer
FF08    RPTR:   DEFS    1               ;Register Pointer
FF0B    TAPESW: DEFS    1               ;Tape speed
FF0E    TABSRP: DEFS    1               ;Tab stops

        ;Transfer Vectors

FF0F    GETGAP: DEFS    3               ;Gap Vector
FF12    KBDTLV: DEFS    3               ;KBDTL Vector
FF15    KBDWFV: DEFS    3               ;KBDWF Vector
```

 Copyright October 1980

```
FF18    VTV:    DEFS    3               ;Console out
FF1B    KBDV:   DEFS    3               ;Console in
FF1E    TOV:    DEFS    3               ;Tape out
FF21    TIV:    DEFS    3               ;Tape in
FF24    LPV:    DEFS    3               ;Line printer out
FF27    OUT1:   DEFS    3
FF2A    OUT2:   DEFS    3
FF2D    IN1:    DEFS    3
FF30    IN2:    DEFS    3
FF33    IN3:    DEFS    3
FF36    TRAPX:  DEFS    3               ;Trap Extension Vector
FF39    MONX:   DEFS    3               ;Monitor Extension Vector

        ;General storage

FF3C    ADDR:   DEFS    2               ;User cold start ADDR
FF3E    UMASK:  DEFS    1               ;UPORT Mask
FF3F    DLYTIM: DEFS    2               ;SIO-2/2B Rate
FF44    LSTATV: DEFS    3               ;LST status vector
FF47    CTBL:   DEFS    CTBLEN+1        ;VT control table
FF6E    INITV:  DEFS    3               ;Disc transfer vectors
FF71    RDSECV: DEFS    3               ;  initialised to 0FFH
FF74    WRSECV: DEFS    3               ;  for cassette
FF77    WRCHKV: DEFS    3
FF87    PATL:   DEFS    1               ;Length of pattern
FF88    PATN:   DEFS    1               ;Start of pattern
FFFE    .IO:    DEFS    2               ;FP Port Pointer
0006 =  HIMEM   EQU     0006H           ;PTR to top of RAM

        ;Port addresses (Memory mapped hardware)

FBFC =  KBD     EQU     0FBFCH          ;Keyboard latch (R)
FBFC =  PORT0   EQU     0FBFCH          ;System control (W)
FBFD =  CNTR    EQU     0FBFDH          ;Tape decode counter (R)
FBFE =  PORT1   EQU     0FBFEH          ;System status (R)
FBFF =  UPORT   EQU     0FBFFH          ;User I/O Port

        ;Port 0 bit allocation (W)

0007 =  PAGE    EQU     7               ;Mem page switch
0006 =  CLRCNT  EQU     6               ;Clear 8 usec counter
0005 =  RLY2    EQU     5               ;Relay 2 (=0 for closed)
0004 =  FREQ    EQU     4               ;Set 2400 Hz
0003 =  RLY1    EQU     3               ;Relay 1 (=0 for closed)
0002 =  VTOPN   EQU     2               ;Open VT memory
0001 =  SINGLE  EQU     1               ;Enable single step (0=set)
0000 =  CLRKBD  EQU     0               ;Clear keyboard latch

        ;Port 1 bit allocation (R)

0007 =  LINE    EQU     7               ;VT Line waveform
0006 =  FRAME   EQU     6               ;VT Frame waveform
0005 =  TAPE    EQU     5               ;Cassette signal sense
0003 =  VOL     EQU     3               ;Cassette volume sense
0002 =  RESET   EQU     2               ;Reset button
0001 =  CLK     EQU     1               ;1200 Hz clock
```

APPENDIX B

CHANGES FROM VERSION 3.0

This section describes the differences between COS versions 3.4 and 3.0.

COS MONITOR COMMAND LEVEL

The M command has been added and enables the High Resolution Graphics board as memory and restarts COS. This will only function correctly on 32K systems.

The O (option) command no longer checks in disc versions for cassette speed control. It goes straight into asking for printer type. If (out of habit) L is typed, the question is typed again. There are now line printer options 5 and 6, indicating the extra serial interfaces SIO-5 and 6, which are additional SIO-4 serial interfaces.

The new X command causes COS to boot CP/M off drive B. This effectively swaps drives A and B, and C and D as far as CP/M is concerned, so all references to A: will access drive B, and vice versa.

The responce to Load and Dump is slightly changed - instead of the address appearing two spaces indented after a flashing cursor, it appears at the beginning of the line. Also, a greater than sign (>) is output after each record which is NOT loaded.

FRONT PANEL

The V command has been added and updates the program counter PC from the memory pointer.

The O command has been changed as described above.

Terminating a command with CTRL L or CTRL O increments or decrements the memory pointer by 32 respectively.

A comma no longer sends a zero to the output port specified by the I/O pointer if there were no hex characters input - instead, the comma is ignored.

VIDEO DISPLAY

The screen handler has been very extensively modified. The cursor is no longer confined to the bottom line of the screen. Instead, various control characters move the cursor round the screen. If the scroller performs an automatic carriage RETURN/LINE FEED because the 40th character has been received, it will ignore the next character if it is a carriage RETURN (or the next two if they are carriage RETURN/LINE FEED). Carriage RETURN acts as a carriage RETURN/LINE FEED, but the next character is ignored if it is a LINE FEED. This behaviour is intended to

 Copyright October 1980

make sure that old programs still work while providing new software with more powerful facilities. The characters interpreted are summarised in Appendix A.

EXTERNAL VDU

COS now checks during intialisation to see if there is a VDU plugged into the SIO-4 socket. If there is, all subsequent output via the scroller output EMTs is sent to this VDU. Graphical output, such as the Front Panel, or low or high resolution graphics, still appear on the RML screen. If a character is received from the VDU at 9600 Baud, COS also switches the keyboard EMTs to take all subsequent input from the VDU ignoring the RML keyboard.

CASSETTE SUPPORT

Slow (300 Baud) cassette support is no longer provided on disc versions, and all cassette I/O in these cases is consequently at 1200 Baud. Appendix C describes a separate utility program which is available to provide slow speeds.

EMT SUPPORT

The EMT and CALR mechanisms have been speeded considerably. This means that certain processes occur rather more quickly than in earlier versions. The minimum execution time for an EMT is now about 65us rather than 125us in earlier versions.

EMTs SETTAB and VERIFY no longer exist, and return immediately to the calling program with registers intact.

On disc versions, EMT SETCAS now simply returns 3, indicating fast input and output, because of the removal of the slow cassette option.

EMT BOOT (code 49 decimal, 31 hex) has been added. This causes CP/M to perform a cold bootstrap, on drive A or B depending on whether the last cold boot was by B or X from command level.

EMT LPSTAT (code 50 decimal, 32 hex) returns A=0 and the zero flag set if the printer device last selected by EMT SETLST is busy, otherwise -1 and non-zero.

EMT VERSN (code 51 decimal, 33 hex) returns the version number of COS in A.

The disc EMTs now go via transfer vectors allowing switching between types of disc.

The EMTs KBDTL, KBDWF and LSTAT also now go via transfer vectors allowing alternative handlers to be installed as required.

APPENDIX C

SLOW CASSETTE SUPPORT

Disc versions of COS 3.4 do not support slow cassette loading and saving at 300 Baud. This means that cassette I/O can only occur at 1200 Baud. A utility program entitled SCASS is available which changes the cassette I/O transfer vectors in COS workspace in order to provide slow cassette transfer rates.

When SCASS is loaded into memory at 100 hex, it relocates itself to high memory and links itself into the cassette transfer vectors. In a cassette system, the I/O handlers themselves run at the top of memory, and the memory location HIMEM is modified to take this into account. On disc systems, the handlers reside in the workspace at FC00 hex. This means that SCASS on disc cannot be run in conjunction with such programs as ADM3A.

Once the slow handlers have been loaded, they continue to be effective until the Reset button is pressed. Until this time, all cassette I/O is at 300 Baud, i.e. it is always slow.

NOTE: The cassette version of SCASS should always be used for cassette software, e.g. BASIC, TXED, etc. The disc version should be used in conjunction with disc software, e.g. BASLOAD, FILEX, etc. SCASS will crash if the wrong version is used.

In disc systems, SCASS should be loaded as follows:

```
A>SCASS


A>CTRL F      (echoed as ¢F)
!CTRL B       (echoed as ¢B)
-
```

Now in Monitor command level, cassette I/O can take place. It should be noted that since SCASS runs from 100 hex, programs to be dumped to cassette should be loaded into memory AFTER SCASS.

In cassette systems, simply use the L command to load SCASS. Again for dumping programs to cassette, they must be stored after SCASS.

```
-L
Name>SCASS
```

 Copyright October 1980

THIS PAGE INTENTIONALLY LEFT BLANK

 Copyright October 1980

Index

Alternate register set 3.2, **3.4**

Bootstrap **2.2**, 4.20
Break Points **3.5**

CALR **6.2**
CP/M 2.1, **2.2**, 6.3
CTRL key **1.2**
Cassette EMTs **4.21**
Cassette Interface **5.4**
Cassette Input 4.21, 4.24, **5.5**
Cassette Output 4.21, 4.24, **5.4**
Cassette Speed **2.5**, 4.22, C.1
Cassette support **2.3**
Character Codes:
From COS Monitor
B Command 2.1, **2.2**
C Command 2.1, **2.6**
D Command 2.1, **2.4**
J Command 2.1, **2.8**
L Command 2.1, **2.3**
M Command 2.1, **2.9**
O Command 2.1, **2.5**
X Command 2.1, **2.2**

From Front Panel
G Command 3.1, **3.6**
H Command 3.1, **3.6**
I Command 3.1, **3.6**
J Command 3.1, **3.6**
K Command 3.1, **3.6**
M Command 3.1, **3.2**
N Command 3.1, **3.6**
O Command 3.1, **3.8**
P Command 3.1, **3.6**
R Command 3.1, **3.2**
S Command 3.1, **3.8**
U Command 3.1, **3.2**
V Command 3.1, **3.2**
X Command 3.1, **3.2**
Z Command 3.1, **3.5**
RETURN 3.1, **3.3**
- Command 3.1, **3.3**
LINEFEED 3.1, **3.3**
/ Command 3.1, **3.3**
. Command 3.1, **3.4**
> Command 3.1, **3.5**
< Command 3.1, **3.5**

Continue execution 2.6, 3.6

Control Codes:
From COS Monitor
CTRL C **2.1**
CTRL F **2.1**
CTRL T 2.1, **2.2**
CTRL Z **2.1**

From Front Panel
CTRL B 3.1, **3.9**
CTRL C 3.1, **3.9**
CTRL L 3.1, **3.3**
CTRL O 3.1, **3.3**

Via EMT OUTC
CTRL D **4.3**
CTRL H **4.3**
CTRL I **4.3**
CTRL J **4.3**
CTRL K 4.3, **4.4**
CTRL L 4.3, **4.4**
CTRL M 4.3, **4.4**
CTRL N 4.3, **4.4**
CTRL O 4.3, **4.4**
CTRL Q 4.3, **4.4**
CTRL S 4.3, **4.4**
CTRL U 4.3, **4.4**
CTRL V 4.3, **4.4**
CTRL W 4.3, **4.5**
CTRL X 4.3, **4.5**
CTRL Y 4.3, **4.5**
CTRL] 4.3, **4.5**
CTRL ^ 4.3, **4.5**
CTRL _ 4.3, **4.5**

Counter Timer Circuits (CTC) **6.2**

DELT key **1.2**
Disc EMTs **4.18**
Disc support **2.2**

EMT Trap Calls **4.1**
EMT BASE 4.11, **4.14**
EMT BOOT 4.18, **4.20**
EMT BYTEO 4.11 **4.12**
EMT CHAN 4.11 **4.12**
EMT CLEAR 4.2, **4.7**
EMT CLOSE 4.2, **4.6**, 5.3
EMT CONTC **4.1**
EMT DEOUT 4.11, **4.12**
EMT ERROR **4.1**
EMT FSTLST 4.11, **4.14**
EMT GETBYT **4.21**, 4.24, 5.5
EMT GETHEX **4.11**
EMT GETJP 4.11, **4.14**
EMT GETSYN 4.21, **4.22**
EMT GRAFIX 4.2, **4.6**

EMT IN1 4.24
EMT IN2 4.24
EMT IN3 4.24
EMT INF 4.2, **4.8**
EMT INIT **4.18**, 4.24
EMT KBDC **4.9**, 4.24
EMT KBDIN **4.9**
EMT KBDTC **4.9**
EMT KBDTF 4.9, **4.10**
EMT KBDW 4.9, **4.10**
EMT KBDWF 4.9, **4.10**, 4.24
EMT LPOUT **4.15**, 4.24
EMT LPSTAT 4.15, 4.17, 4.24
EMT MSG 4.2, **4.7**
EMT OPNWT 4.2, **4.6**, 5.3
EMT OUT1 4.24
EMT OUT2 4.24
EMT OUTC **4.2**, 4.24
EMT OUTC(2) 4.2, **4.7**
EMT OUTF 4.2, **4.8**
EMT OUTNC 4.2, **4.8**
EMT PUTBYT **4.21**, 4.24, 5.5
EMT RDSEC **4.18**, 4.24
EMT S4KIN 4.15, **4.17**
EMT S4KTL 4.15, **4.17**
EMT SAVE 4.11, **4.13**
EMT SAVEA 4.11, **4.13**
EMT SETCAS 4.21, **4.22**
EMT SETLST **4.15**
EMT SCROLL 4.2, **4.6**
EMT UPDATE **4.11**
EMT VERSN 4.11, **4.15**
EMT WAIT **4.11**
EMT WRCHK 4.18, **4.20**, 4.24
EMT WRSEC 4.18, **4.19**, 4.24

Fill memory **3.7**
Flags **3.2**
Frame Output **5.3**
Front Panel display **3.1**
Front panel entry **3.9**

GETGAP 4.22 **4.25**

HIMEM **6.3**
Hexadecimal calculator **3.7**
Hexadecimal numbers 1.2
High Resolution Graphics board 2.9

I/O Channels **4.24**
I/O Ports 3.2, **3.4**
I/O Transfer Vectors **4.25**
I/O Transfers **4.24**
Initialisation **6.1**
Interrupts **6.2**

Kansas City Standard (CUTS) **5.5**, 5.6
Keyboard 1.2, **5.1**
Keyboard EMTs **4.8**

Line output **5.3**
Load errors **2.3**
Loading from cassettes **2.2**

MONX **4.25**
Memory pointers **3.2**
Memory patterns **3.6**
Modifying memory **3.2**
Modifying registers **3.4**

Position Independent Code **6.1**
Printer EMTs **4.15**
Printer support **2.6**
Printer types 2.5, **2.6**, 4.15

Quick Reference guide A.1

RETURN key 1.2, 3.1, **3.3**
Reset **6.1**
Restart Addresses **2.5**

Saving on Cassettes **2.4**
Screen EMTs **4.2**
Screen timing **5.2**
Shift memory **3.8**
Single Step **3.5**

TRAPX **4.26**
Test memory 3.7

User I/O port **5.1**

Version numbers **2.1**, 4.15
Video Display **5.2**

Z80 Registers **3.2**, 3.4

IPN 01158101